INFORMATION TECHNOLOGY LAW

“MICRO-AGREEMENTS” IN SYSTEMS INTEGRATION AND OUTSOURCING PROJECTS

Recognising and managing the legal implications of day to day interactions between parties to large and complex information technology projects

PAUL P. PARNELL
February 2000

Prepared in satisfaction of requirements for the Master of Laws (Honours) Degree at the University of Western Sydney, Nepean
PLEASE NOTE

The greatest amount of care has been taken while scanning this thesis,

and the best possible result has been obtained.
Synopsis

Information Technology Law: "Micro-agreements" in systems integration and outsourcing projects - Recognising and managing the legal implications of day to day interactions between parties to large and complex information technology projects

This work describes the new concept of a "micro-agreement". Micro-agreements represent the many forms of interaction, which occur between the parties involved in large and complex information technology projects. Micro-agreements can provide benefits as well as disadvantages to such projects and need to be effectively managed.

This work begins by describing the nature of information technology projects from an engineering perspective, particularly in light of the problems that may occur. The existing legal doctrines that are relevant to such projects are then described and these doctrines are expanded into the new concept of a "micro-agreement". The concept of "micro-agreements" is supported through the analysis of a number of case studies relevant to the information technology industry, together with further analysis of legal relationship models.

Overall, a number of key recommendations are made which provide support for gaining maximum benefits from micro-agreements. These recommendations include firstly, linking information technology contracts to software engineering “best practice”; secondly, using an appropriate legal relationship model; and thirdly, developing an industry wide Information Technology Code of Conduct.
Overview of Contents

1. INTRODUCTION AND OBJECTIVES ................................................................. 1

2. THE NATURE OF LARGE INFORMATION TECHNOLOGY PROJECTS .................. 3

3. CONTRACT INTERPRETATION AND RELATED ISSUES .................................. 29

4. “MICRO-AGREEMENTS” PART 1: CONCEPTS AND CASE STUDIES .................... 82

5. “MICRO-AGREEMENTS” PART 2: RELATIONSHIP MODELS ............................. 183

6. CONCLUSION .................................................................................................. 273
Detailed Contents

1. INTRODUCTION AND OBJECTIVES.............................................................. 1

2. THE NATURE OF LARGE INFORMATION TECHNOLOGY PROJECTS.............. 3

   2.1 INTRODUCTION ................................................................................. 3
   2.2 BACKGROUND ................................................................................. 3
   2.2.1 INTRODUCTION ........................................................................... 3
   2.2.2 HOW LARGE IS LARGE? ................................................................. 4
   2.3 SYSTEMS ENGINEERING ................................................................. 6
   2.3.1 INTRODUCTION ........................................................................... 7
   2.3.2 A DEFINITION ............................................................................ 7
   2.3.3 ENGINEERING "TRADE-OFFS" ................................................. 9
   2.4 THE SOFTWARE LIFECYCLE ............................................................ 9
   2.4.1 INTRODUCTION ........................................................................... 9
   2.4.2 LIFECYCLE MODELS ................................................................. 10
   2.4.2.1 The Waterfall Model of Software Development .......... 10
   2.4.2.2 Iterative and Evolutionary Software Development Models 11
   2.4.3 CHANGE ..................................................................................... 13
   2.5 REQUIREMENTS MANAGEMENT .................................................. 14
   2.5.1 WHAT ARE REQUIREMENTS? ..................................................... 14
   2.5.2 THE PROBLEM .......................................................................... 14
   2.5.3 WHY DO REQUIREMENTS CHANGE? ....................................... 15
   2.5.4 HOW MUCH DO REQUIREMENTS CHANGE? .......................... 16
   2.5.5 HOW TO MINIMISE CHANGES IN REQUIREMENTS .................. 18
   2.5.5.1 Introduction ........................................................................... 18
   2.5.5.2 Joint Application Development .......................................... 19
   2.5.5.3 Prototyping ........................................................................... 19
   2.5.5.4 Inspections ........................................................................... 20
   2.5.5.5 Cost-per-Function-Point Contracts .................................. 20
   2.5.5.6 Change-Control Boards ...................................................... 20
   2.5.6 IS REQUIREMENTS ENGINEERING RELEVANT? .................... 21
   2.5.7 CONCLUSION ............................................................................ 22
   2.6 METRICS AND MEASUREMENT .................................................... 22
   2.6.1 INTRODUCTION ........................................................................... 22
   2.6.2 WHY MEASURE SOFTWARE? ................................................... 23
   2.6.3 SOME SPECIFIC SOFTWARE MEASURES ............................... 24
   2.6.3.1 Complexity Metrics .............................................................. 24
   2.6.3.2 Function Point Metrics ......................................................... 24
   2.6.3.3 Some Areas have no Reliable Metrics ................................. 25
   2.6.4 “EXTERNAL” SOFTWARE METRICS ....................................... 25
   2.6.5 CONCLUSION ............................................................................ 25
   2.7 DEFECTS ......................................................................................... 26
   2.7.1 INTRODUCTION ........................................................................... 26
   2.7.2 HOW MANY DEFECTS OCCUR? ................................................. 26
   2.8 SYSTEMS INTEGRATION CONTRACTS ........................................ 27
   2.9 CONCLUSION .................................................................................. 28

3. CONTRACT INTERPRETATION AND RELATED ISSUES ............................ 29

   3.1 INTRODUCTION .............................................................................. 29
   3.2 THE PRE-CONTRACTUAL ENVIRONMENT ................................... 29

Paul P. Parnell
3.4.10 IMPLYING A TERM WHERE ONE PARTY HAS ALREADY PERFORMED OBLIGATIONS ACCORDING TO THAT TERM .......... 75
3.4.11 DUTY TO CO-OPERATE .................................................................................................................. 75
3.4.12 TERMS IMPLIED BY STATUTE ..................................................................................................... 75
3.4.12.1 Trade Practices Act ..................................................................................................................... 75
3.4.12.2 Sale of Goods Act ....................................................................................................................... 78
3.4.12.3 Other Legislation ....................................................................................................................... 79

3.5 UNJUST ENRICHMENT, QUANTUM MERUIT, MONEY HAD AND RECEIVED 79
3.5.1 QUANTUM MERUIT ..................................................................................................................... 79
3.5.2 MONEY HAD AND RECEIVED ...................................................................................................... 80

3.6 CONCLUSION .................................................................................................................................. 80

4. "MICRO-AGREEMENTS" PART 1: CONCEPTS AND CASE STUDIES ......................................................... 82

4.1 INTRODUCTION .................................................................................................................................. 82
4.2 METHOD OF ANALYSIS AND EVALUATION ..................................................................................... 83
4.3 DEFINITIONS ..................................................................................................................................... 83
4.3.1 MICRO-AGREEMENT ..................................................................................................................... 83
4.3.2 SYSTEMS INTEGRATION ................................................................................................................. 84
4.3.3 SOFTWARE ENGINEERING ........................................................................................................... 84
4.4 LINKING THE SYSTEMS INTEGRATION CONTRACT TO SOFTWARE ENGINEERING "BEST PRACTICE" 85
4.4.1 INTRODUCTION ............................................................................................................................. 85
4.4.2 WHAT IS SOFTWARE ENGINEERING "BEST PRACTICE"? .............................................................. 85

4.5 EXAMPLES AND CASE STUDIES - SYSTEMS INTEGRATION MANAGEMENT 90

4.5.1 INTRODUCTION AND EXPLANATION OF THE ACTIVITY ......................................................... 91
4.5.2 EXAMPLE OF A MICRO-AGREEMENT ......................................................................................... 91
4.5.3 POTENTIAL PROBLEMS ARISING FROM THE MICRO-AGREEMENT ......................................... 93
4.5.4 AVOIDING THE PROBLEM ........................................................................................................... 94
4.5.5 LEGAL ANALYSIS OF PROBLEMS .............................................................................................. 97
4.5.5.1 Summary of Legal Claims ........................................................................................................... 97
4.5.5.2 Misrepresentation - Common Law ............................................................................................... 99
4.5.5.3 Misrepresentation under the Trade Practices Act (1974) .......................................................... 101
4.5.5.4 Breach of Contract .................................................................................................................... 103
4.5.5.5 Estoppel ................................................................................................................................... 106
4.5.5.6 Economic Duress ....................................................................................................................... 109
4.5.5.7 Unconscionable Transactions ................................................................................................... 110
4.5.6 SUGGESTED IMPROVEMENTS IN THE LAW ............................................................................... 112
4.5.6.1 Introduction ............................................................................................................................... 112
4.5.6.2 Implied Contractual Duty to Co-operate .................................................................................. 112

4.5.7 INDUSTRY CODE OF CONDUCT ................................................................................................. 113
4.5.8 FINDINGS .................................................................................................................................... 113
4.5.9 RECOMMENDATIONS ................................................................................................................... 115
4.5.10 CONCLUSION ............................................................................................................................... 117

4.6 CASE STUDY 2 - MICRO-AGREEMENTS IN THE CONTEXT OF SOFTWARE PROJECT PLANNING 118

4.6.1 INTRODUCTION AND EXPLANATION OF THE ACTIVITY ......................................................... 118
4.6.2 EXAMPLE OF A MICRO-AGREEMENT ......................................................................................... 118
4.6.3 POTENTIAL PROBLEMS ARISING FROM THE MICRO-AGREEMENT ......................................... 119
4.6.4 AVOIDING THE PROBLEM ........................................................................................................... 119
4.6.5 LEGAL ANALYSIS OF PROBLEMS .............................................................................................. 122
4.6.5.1 Quasi-contract, Unjust Enrichment and Quantum Meruit .......................................................... 122
4.6.6 INDUSTRY CODE OF CONDUCT ................................................................................................. 123
4.6.7 FINDINGS .................................................................................................................................... 123
4.6.8 RECOMMENDATIONS ................................................................................................................... 124
4.6.9 CONCLUSION ............................................................................................................................... 125
4.12 SUMMARY OF FINDINGS ................................................. 173
4.12.1 INTRODUCTION ......................................................... 173
4.12.2 LIST OF CASE STUDY FINDINGS ..................................... 174
4.12.3 CONSOLIDATED FINDINGS ........................................... 177
4.13 SUMMARY OF RECOMMENDATIONS ................................. 177
4.13.1 INTRODUCTION ......................................................... 177
4.13.2 LIST OF CASE STUDY RECOMMENDATIONS ..................... 177
4.13.3 CONSOLIDATED RECOMMENDATIONS .............................. 181
4.14 CONCLUSION .............................................................. 182

5. "MICRO-AGREEMENTS" PART 2: RELATIONSHIP MODELS ................. 183

5.1 INTRODUCTION ............................................................. 183
5.2 LEGAL RELATIONSHIPS .................................................... 185
  5.2.1 INTRODUCTION ......................................................... 185
  5.2.2 PARTNERSHIP .......................................................... 185
  5.2.2.1 Definition ........................................................... 185
  5.2.2.2 Basic Principles .................................................. 185
  5.2.2.3 Liability ............................................................ 187
  5.2.2.4 Partnership and Agency ......................................... 188
  5.2.2.5 Advantages ........................................................ 188
  5.2.2.6 Disadvantages ..................................................... 189
  5.2.3 JOINT VENTURE ....................................................... 189
  5.2.3.1 Definition ........................................................... 189
  5.2.3.2 Principles .......................................................... 190
  5.2.3.3 Advantages ........................................................ 190
  5.2.4 TRUSTS ................................................................. 190
    5.2.4.1 Definition and Types of Trust ................................. 191
    5.2.4.2 Express Trusts ................................................... 191
7.5.2.4.3 Non-Express Trusts ............................................ 192
7.5.2.4.4 Fiduciary Obligations ......................................... 193
7.5.2.4.5 Advantages ........................................................ 195
7.5.2.4.6 Disadvantages ..................................................... 195
5.3 ENGINEERING AND BUSINESS RELATIONSHIPS .................. 196
  5.3.1 JOINT APPLICATION DEVELOPMENT .............................. 196
  5.3.1.1 Definition ........................................................... 196
  5.3.1.2 Principles .......................................................... 196
  5.3.1.3 Advantages ........................................................ 196
  5.3.2 PARTNERING .......................................................... 197
  5.3.2.1 Definition ........................................................... 197
  5.3.2.2 Principles .......................................................... 197
  5.3.2.3 Advantages ........................................................ 198
  5.3.2.4 Disadvantages ..................................................... 198
    5.3.2.5 Conclusion ......................................................... 199
5.4 RELATIONSHIP MODELS .................................................. 200
  5.4.1 INTRODUCTION ......................................................... 200
  5.4.2 METHOD OF ANALYSIS OF RELATIONSHIP MODELS ............ 201
5.5 CONVENTIONAL RELATIONSHIP MODEL ............................. 204
  5.5.1 DESCRIPTION .......................................................... 204
  5.5.2 EXAMPLE ............................................................... 204
  5.5.3 ANALYSING THIS RELATIONSHIP MODEL ......................... 204
    5.5.3.1 Micro-agreements in the context of Requirements Management 204
    5.5.3.2 Micro-agreements in the Context of Software Project Planning 210
    5.5.3.3 Micro-agreements in the Context of Software Project Tracking and Oversight 211
    5.5.3.4 Micro-agreements in the Context of Sub-Contractor Management 213
    5.5.3.5 Micro-agreements in the Context of Software Quality Assurance 215
    5.5.3.6 Micro-agreements in the Context of Software Configuration Management 217
    5.5.3.7 Assessment of Legal Risks ..................................... 217
A work of this size and complexity can not be produced entirely in isolation. Accordingly, I cannot lose this opportunity to acknowledge the significant contribution of a number of people.

The prime motivating force for this work came from within the Faculty of Law at the University of Western Sydney, Nepean. Specifically, my appreciation goes to my principal research supervisor, Dr. Olujoke E.A. Akindemowo, for her totally professional approach to the law, her persistence in reviewing my drafts, and for her constructive criticism and support generally. Dr. Akindemowo provided many suggestions and recommendations throughout this project and her expertise and professionalism is greatly appreciated.

I would also take the opportunity to thank Professor Tuan Razeen Sappideen from the University of Western Sydney, Nepean for his general support, guidance, and encouragement for this work, particularly during the early stages of the project.

My thanks also to the support staff at the UWS Faculty of Law for their efficiency and dedication in dealing with administrative matters.

Last but by no means least, I would like to thank my Wife Ann for her support, encouragement, and understanding, and as a legal specialist in her own right, her review of my work. An undertaking of this size and complexity can be made significantly less arduous and more enjoyable when it is done with the support of family and I sincerely thank Ann in that regard.
### Table of Cases

<table>
<thead>
<tr>
<th>CASES</th>
<th></th>
<th>34, 101</th>
<th>33</th>
<th>74</th>
<th>50</th>
<th>38</th>
<th>39, 110</th>
<th>80, 133</th>
<th>123, 164</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alati v Kruger (1955) 94 CLR 216</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Allcard v Skinner (1887) 36 Ch.D. 145</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Allstate Life Insurance Co. and Others v Australian and New Zealand Banking Group Limited and Others</td>
<td>No. G381 of 1994 FED No. 45/95</td>
<td>164</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ansett Transport Industries v Commonwealth (1977) 199 CLR 54</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aotearoa International Ltd. v Scancarriers A/S (1985) 1 N.Z.L.R. 515 at 556</td>
<td></td>
<td>68</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Argy Trading Development Co Ltd v Lapid Developments Ltd (1977) 1 WLR 444</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Astley v Reynolds (1731) 93 E.R. 939</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Atlas Express Ltd. v Keco (Importers &amp; Distributors) Ltd. (1989) 3 W.L.R. 389</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Atwood v Small (1838) 6 Cl &amp; Fin. 232; 7 E.R. 684</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Austotel Limited v Franklins (1989) 16 NSWLR 582</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>107</td>
</tr>
<tr>
<td>Australia and New Zealand Banking Group Ltd. v Westpac Banking Corp. (1988) 62 ALJR 293</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Australian &amp; New Zealand Banking Group Ltd v Westpac Banking Corporation (1988) 62 ALJR 29279, 123</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Australian Oil &amp; Gas Corporation Limited v Bridge Oil Limited, NSW Court of Appeal, Kirby P, 12 April 1989, unreported</td>
<td></td>
<td>190</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B.P. Refinery (Westernport) Pty Ltd v Shire of Hastings (1977) 16 ALR 363</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B.P. Refinery (Westernport) Pty Ltd v Shire of Hastings (1977) 52 ALJR 20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bank of New Zealand v Simpson (1900) A.C. 182</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Barba v Fule Corp of Victoria (1976) 136 CLR 120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Barton v Armstrong (1973) 2 NSWLR 598 at 606</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Barton v Armstrong (1976) A.C. 104</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bateman v Slater (1987) 71 ALR 55</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Beattie v Lord Ebury (1872) L.R. 7 Ch. App. 777</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Beta Computers (Europe) Ltd v Adobe Systems Ltd 1996 SLT 604; 1996 SCLR 587</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bisset v Wilkinson (1927) A.C. 177</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bloch v Bloch (1981) 37 ALR 55</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blomley v Ryan (1956) 99 CLR 362</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blomley v Ryan (1956) 99 CLR 362</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bohemians Club v Acting Federal Commissioner of Taxation (1918) 24 CLR 334</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bond Corporation Pty Ltd v Thiess Contractors Pty Ltd (1987) 14 FCR 215</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bradley Egg Farm v Clifford (1943) 2 All ER 378</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Breen v Williams (1996) 186 CLR 71</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Brian Pty Ltd v United Dominions Corporation Ltd. (1983) 1 NSWLR 490</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Brikom Investments Ltd v Carr (1979) QB 467</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>British Movietoners Ltd. v London &amp; District Cinemas Ltd (1952) AC 166</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>British Steel Corporation v Cleveland Bridge and Engineering Company Ltd (1984) 1 All ER 504, 80, 123</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Brown v Jam Factory Pty. Ltd. (1981) 35 ALR 79 at 88</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Brown v Raphael (1958) Ch. 636</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byrne v Australian Airlines Ltd (1995) 131 ALR 422</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byrne v Australian Airlines Ltd. (1995) 131 ALR 422</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Car &amp; Universal Finance Co. Ltd. v Caldwell (1965) 1 Q.B. 525</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Caslec Industries Pty Ltd v Windhover Data Systems Pty Ltd (Federal Court of Australia, Gummow J, No G627/90, 13 Aug 1992, unreported)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cassels v Stewart (1881) 6 App. Cas. at 79</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Castlemaine Tooweps Ltd. v Carlton &amp; United Brewers Ltd. (1987) 10 NSWLR 468</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Central London Property Trust Ltd. v High Trees House Ltd. (1947) K.B. 130</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Checker Taxicab Ltd v Stone (1930) NZLR 169</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Codeca Construction Pty Ltd v State Rail Authority of New South Wales (1982) A.L.J.R. 459</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Codeca Construction Pty Ltd v State Rail Authority of NSW (1982) 149 C.L.R. 33</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Codeca Construction Pty Ltd v State Rail Authority of NSW (1982) 149 CLR 337</td>
<td></td>
<td>64, 66, 67, 104</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Codeca Construction Pty Ltd v State Rail Authority of NSW (1982) 56 A.L.J.R. 459</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Collins Marrickville Pty Ltd v Henjo Investments Pty Ltd (1987) 72 ALR 601</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Paul P. Parnell  xii
Combe v Combe .......................................................... 52
Combe v Combe [1951] 2 KB 215 ........................................ 50
Commercial Bank of Australia v Amadio (1983) 151 CLR 447 .......................... 41, 42, 44
Commercial Bank of Australia v Amadio (1983) 151 CLR 447 .......................... 42
Commercial Banking Company (Sydney) v R.H.Brown & Co. (1972) 126 C.L.R. 337 ........................................ 33
Commonwealth v Sciutte Savings Bank (1884) 137 Mass. 301 ........................... 52
Commonwealth v Verwayen (1990) 170 CLR 394 ............................... 48
Commonwealth v Verwayen (1990) 170 CLR 394 ............................... 48
Commonwealth v Verwayen (1990) 64 ALJR 540 ................................... 54, 109
Concrete Constructions (NSW) Pty Ltd v Nelson (1990) 169 CLR 594 ............ 101
Con-Stan Industries of Australia Pty. Ltd v Norwich Winterthur Insurance (Australia) Limited (1986) 160 C.L.R. 226 ................................................................. 69, 104
Couchman v Hill [1947] K.B. 554 ................................................ 60
Coulls v Bagot's Executor and Trustee Co Ltd (1967) CLR 460 ........................ 105
Crabb v Aron District Council (1976) Ch. 179 ........................................... 54
Craven-Ellis v Canons Ltd (1936) 2 KB 403 ......................................... 79, 123, 164
Crawford Fitting Co v Valve & Fittings Pty Limited (1988) 14 NSWLR 438 ..... 73
Crescendo Management Pty. Ltd v Westpac Banking Corp. (1988) 19 N.S.W.L.R. 40 .................................................. 39, 110
Crosse v Gardner (1689) Carth. 90; 90 E.R.656 ........................................ 59
D & C Builders Ltd v Rees [1966] 2QB 617 ........................................... 50
D. & C. Builders Ltd v Rees [1966] 2QB 617 ........................................... 50
D.J. Hill & Co Pty Ltd v [1971] VR 749 ............................................ 71
David Securities Pty Ltd v Commonwealth Bank of Australia (1992) 175 CLR 353 .................................................. 32
Davies v Provincial Marine Insurance Co. (1878) 8 Ch.D. 469 ........................ 30
Derry v Peek (1889) 14 App. Cas. 337 ............................................ 33
Dick Bentley Productions Ltd v Harold Smith (Motors) Ltd [1965] 2 All E.R. 65 . 35
Dimmock v Hallert (1866) L.R. 2 Ch. 21 ........................................... 30
Dyer v Dyer (1788) 2 Cox 92 .................................................... 195
Edgington v Fitzmaurice (1885) 29 Ch.D. 459 ......................................... 31, 100
Eggleson v Marley Engineers Pty Ltd (1979) 21 SASR 51 .............................. 71
Ellul and Ellul v Oakes (1972) 3 S.A.S.R. 377 ........................................ 35
Esso Australia Resources Ltd v Plowman (1995) 69 ALJR 404 ...................... 70, 72
Esso Australia Resources Ltd v Plowman (1995) 131 ALR 422 ....................... 66
Foley v Classique Coaches Ltd [1934] 2 K.B. ........................................... 74
Foran v Wight (1989) 168 CLR 385 ............................................. 148
Foran v Wight (1990) 64 ALJR 1 ................................................... 53, 109
Fraser v Thames Television Ltd. [1984] Q.B. 44 ....................................... 66
French v Syring (1857) 140 ER. 455 ................................................ 187
Georgis v European Asia Bank (1989) .............................................. 41
Gordon v Gordon (1821) 3 Swans. 400; 36 E.R. 910 ................................ 30
Great Western Rly v Bristol Corp. (1918) 87 L.I. Ch. 414 ........................... 63
Gregg v Tasmanian Trustees Ltd. (1997) 143 ALR 328 .............................. 44
Grunt v Great Boulder Pty Gold Mines Ltd (1937) C.L.R. 641 ....................... 46, 55, 106, 147
Hawkins v Clayton (1988) 164 CLR 539 ............................................. 65, 104
Hedley Byrne & Co. Ltd v Heller & Partners Ltd. [1964] AC 465 ............... 32
Helmore v Smith (1886) 35 Ch.D. 436 ................................................ 31
Henjo Investments Pty Ltd v Collins Marrickville Pty Ltd (1988) 79 ALR 83 ...... 102
Hick v Raymond & Reid [1893] AC 22 ............................................. 72
Holt v Biroka Pty. Ltd. (1988) 13 NSWLR 629 ....................................... 37
Hornby Building Information Centre v Sydney Building Information Centre (1978) 140 CLR 216 .................................................. 37
Horsfall v Thomas (1862) 1 H.&C 90; 158 E.R. 813 ................................ 33
Hospital Products Ltd v United States Surgical Corporation (1984) 156 CLR 41 .................................................. 195
Hospital Products Ltd v United States Surgical Corporation (1984) 156 CLR 41 .................................................. 64, 72, 104
Hospital Products v United States Surgical Corporation (1984) 156 CLR 41 .................................................. 134, 194
Hospital Products v United States Surgical Corporation (1984) 156 CLR 41 .................................................. 66
Hoyt's Pty Ltd v Spencer (1919) 27 C.L.R. 133 ....................................... 56, 57
Hughes v Metropolitan Railway Co. (1877) 2 App. Cas. 459 ....................... 48, 51
Hutton v Warren [1936] 1 N & W 466 ................................................ 63, 103
In re Brocklehurst deceased; Hall v Roberts [1978] Ch 14 ....................... 41

Paul P. Parnell
San Sebastian Pty. Ltd. v Minster Administering the Environmental Planning and Assessment Act 1979
(1986) 162 CLR 340 ........................................... 100
Saphena Computing Ltd v Allied Collection Agencies Ltd [(1995) FSR 616 ......................... 205
Saphena Computing Ltd v Allied Collection Agencies Ltd [1995] FSR 616 ......................... 77
Seddon v North Eastern Salt Co. [1905] 1 Ch. 326 .............................................................. 36
Shell U.K. Ltd. v Lostock Garage Ltd. [1976] 1 W.L.R. 1187 ............................................. 68
Shirlaw v Southern Foundaries (1926) Ltd [1939] 2 K.B. 206 ......................................... 67
Silovi Pty Ltd v Barbaro (1988) 13 N.S.W.L.R. 466 .................................................. 47, 48
Skate v Beale (1841) 11 Ad.&E. 983 ................................................................. 37
Smith v Anderson (1880) 15 Ch D 247 ............................................................... 187
Smith v Chadwick (1884) 9 App. Cas. 187 ................................................................. 33
Smith v Land & House Property Corp (1884) 28 Ch. D. 7 ........................................... 31
Smith v William Charlick Ltd (1924) 34 C.L.R. 38 ................................................... 37
Smith v William Charlick Ltd (1924) 34 CLR 38 ...................................................... 39
Spiro v Lintern [1973] 1 WLR 1002; [1973] 3 All ER 319 .............................................. 48
St. Albans City Council v ICL (Q.B., Baker J, 3 October 1994, unreported) ................. 77
State Rail Authority of New South Wales v Heath Outdoor Pty Ltd (1986) 7 N.S.W.L.R. 170 ........ 62
State Rail Authority of New South Wales v Heath Outdoor Pty. Ltd. (1986) 7 NSWR 170 .......... 54, 107
Stewart v Raval's Garage [1952] 2 Q.B. 545 ................................................................. 74
Sutton v A.J. Thompson Pty Ltd (In L)g (1987) 73 ALR 233 ...................................... 102
Sydney City Council v West (1965) 114 CLR 481 ...................................................... 71
Sykes (Wessex) Ltd. v Fine Fare Ltd. [1967] 1 Lloyd's Rep. 53 .................................. 75
Taco Company of Australia v Taco Bell Pty. Ltd. (1982) 57 ALJR 109 ......................... 37
Tamplin (F.A.) Steamship Co. Ltd v Anglo-Mexican Petroleum Products Co. Ltd. [1916] 2 A.C. 397 .... 68
The Moorcock (1889) 14 P.D. 64 ................................................................................. 66
Thomas National Transport (Melbourne) Pty. Ltd. v May & Baker (Australia) Pty. Ltd. (1966) 115 CLR 353 ............... 71
Toby Constructions Products Pty Ltd v Computa Bar (Sales) Pty Ltd (1983) 2 NSWR 48 ........... 77
Tool Metal Manufacturing Co Ltd v Tungsten Electric Co Ltd [1955] 2 All ER 657 ............ 50
Turnbull v Ah Moy (1871) 2 Australian Jurist Reports 40 ........................................... 187
United Dominion Corporation Ltd. v Brian Pty Ltd (1985) 157 CLR 1 .......................... 186
United Dominions Corporation Ltd. v Brian Pty Ltd (1985) 59 ALJR 676 .................... 190
Universe Tankships [1983] 1 A.C. 336 .................................................................... 38
Walker v European Electronics Pty Ltd (in lig) (1990) 23 NSWLr 1 ............................. 188
Walton's (Interstate) v Maher (1988) 62 ALJR 110 ..................................................... 53
Waltons Stores (Interstate) Ltd. v Maher & Anor. (1988) 62 ALJR 110 ......................... 46
Wardley Australia Ltd. v Western Australia (1992) 175 CLR 514 ............................... 102
Wells (Merstham) Ltd v Buckland Sand And Silica Ltd [1965] 2 Q.B. 170 ................. 58, 59
Westmelton (Vic) Pty. Ltd. v Archer and Schulman [1982] V.R. 305 ......................... 41
Westsub Discounts Pty Limited v Idaps Australia Limited (1990) 17 IPR 185 ................. 144, 145, 146
Westsub Discounts v IDAPS Australia Limited (1990) 17 IPR 185 ............................. 144, 145, 146
White Rose Flour Milling Co Pty Ltd v Australian Wheat Board (1944) 18 ALJR 324 .......... 39, 40
White v Bluet (1853) 23 LJ (NS) Ex 36 ................................................................. 105
Wilde v Gibson (1848) 1 HLC 605 ............................................................................. 36
Willis-Smith v Voitno Bros Pty Ltd, unrep., Federal Court of Australia, Lee J. 3 Nov 1944 .... 42
Wise v Perpetual Trustee Co. Ltd. [1903] AC 139 ...................................................... 187
WJ Alan & Co Ltd v El Nasr Export & Import Co [1972] 2 QB 189 ......................... 50
Woodhouse AC Israel Cocoa Ltd SA v Nigerian Produce Marketing Co Ltd[1972] AC 741 .... 50
Yorkville Nominees Pty Ltd (In L)g v Lissenden (1986) 160 CLR 745 ......................... 31
Young and Marten Ltd. v McManus Childs Ltd. [1969] 1 A.C. 454 .................................. 65
Zorba Structural Steel Company Pty Ltd v Walco (1993) 115 FLR 296 ......................... 67

Paul R. Donald

XV
# Table of Statutes

<table>
<thead>
<tr>
<th>Statute</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consumer Affairs and Fair Trading Act 1990 (NT) Section 42</td>
<td>36</td>
</tr>
<tr>
<td>Consumer Affairs and Fair Trading Act 1990 (NT) Section 43</td>
<td>45</td>
</tr>
<tr>
<td>Contracts Review Act, 1980 Section 4(1)</td>
<td>46</td>
</tr>
<tr>
<td>Conveyancing Act 1919 (NSW)</td>
<td>79</td>
</tr>
<tr>
<td>Corporations Law Section 111A</td>
<td>186</td>
</tr>
<tr>
<td>Corporations Law Section 112(2)</td>
<td>186</td>
</tr>
<tr>
<td>Fair Trading Act (N.S.W.) s 43</td>
<td>45</td>
</tr>
<tr>
<td>Fair Trading Act 1985 (Vic), Section 11</td>
<td>36</td>
</tr>
<tr>
<td>Fair Trading Act 1985 (Vic), Section 11A</td>
<td>45</td>
</tr>
<tr>
<td>Fair Trading Act 1997 (NSW), Section 42</td>
<td>36</td>
</tr>
<tr>
<td>Fair Trading Act 1997 (NSW), Section 68</td>
<td>37</td>
</tr>
<tr>
<td>Fair Trading Act 1997 (NSW), Section 72(5)(a)</td>
<td>37</td>
</tr>
<tr>
<td>Fair Trading Act 1997 (NSW), Section 72(5)(b)</td>
<td>37</td>
</tr>
<tr>
<td>Fair Trading Act 1997 (NSW), Section 72(5)(c)</td>
<td>37</td>
</tr>
<tr>
<td>Fair Trading Act 1987 (SA), Section 56</td>
<td>36</td>
</tr>
<tr>
<td>Fair Trading Act 1987 (SA), Section 57</td>
<td>45</td>
</tr>
<tr>
<td>Fair Trading Act 1987 (WA), Section 10</td>
<td>36</td>
</tr>
<tr>
<td>Fair Trading Act 1987 (WA), Section 11</td>
<td>45</td>
</tr>
<tr>
<td>Fair Trading Act 1989 (Qld), Section 38</td>
<td>36</td>
</tr>
<tr>
<td>Fair Trading Act 1989 (Qld), Section 39</td>
<td>45</td>
</tr>
<tr>
<td>Fair Trading Act 1990 (Tas), Section 14</td>
<td>36</td>
</tr>
<tr>
<td>Fair Trading Act 1990 (Tas), Section 15</td>
<td>45</td>
</tr>
<tr>
<td>Fair Trading Act 1992 (ACT), Section 12</td>
<td>36</td>
</tr>
<tr>
<td>Fair Trading Act 1992 (ACT), Section 13</td>
<td>45</td>
</tr>
<tr>
<td>Goods Act 1958 (Vic)</td>
<td>78</td>
</tr>
<tr>
<td>Income Tax Assessment Act, 1936</td>
<td>191</td>
</tr>
<tr>
<td>Industrial Arbitration Act, Section 88F</td>
<td>45</td>
</tr>
<tr>
<td>Partnership Act (ACT) section 6</td>
<td>185</td>
</tr>
<tr>
<td>Partnership Act (ACT), section 13</td>
<td>187</td>
</tr>
<tr>
<td>Partnership Act (ACT), section 14</td>
<td>188</td>
</tr>
<tr>
<td>Partnership Act (ACT), section 16</td>
<td>188</td>
</tr>
<tr>
<td>Partnership Act (ACT), section 29</td>
<td>187</td>
</tr>
<tr>
<td>Partnership Act (ACT), section 5</td>
<td>185</td>
</tr>
<tr>
<td>Partnership Act (NSW) section 1</td>
<td>185</td>
</tr>
<tr>
<td>Partnership Act (NSW), section 10</td>
<td>188</td>
</tr>
<tr>
<td>Partnership Act (NSW), section 12</td>
<td>188</td>
</tr>
<tr>
<td>Partnership Act (NSW), section 24</td>
<td>187</td>
</tr>
<tr>
<td>Partnership Act (NSW), section 45</td>
<td>186</td>
</tr>
<tr>
<td>Partnership Act (NSW), section 46</td>
<td>185</td>
</tr>
<tr>
<td>Partnership Act (NSW), section 9</td>
<td>187</td>
</tr>
<tr>
<td>Partnership Act (Qld), section 12</td>
<td>187</td>
</tr>
<tr>
<td>Partnership Act (Qld), section 13</td>
<td>188</td>
</tr>
<tr>
<td>Partnership Act (Qld), section 15</td>
<td>188</td>
</tr>
<tr>
<td>Partnership Act (Qld), section 27</td>
<td>187</td>
</tr>
<tr>
<td>Partnership Act (Qld), section 3</td>
<td>186</td>
</tr>
<tr>
<td>Partnership Act (Qld), section 48</td>
<td>185</td>
</tr>
<tr>
<td>Partnership Act (Queensland) section 5</td>
<td>185</td>
</tr>
<tr>
<td>Partnership Act (SA) section 1</td>
<td>185</td>
</tr>
<tr>
<td>Partnership Act (SA), section 10</td>
<td>188</td>
</tr>
<tr>
<td>Partnership Act (SA), section 12</td>
<td>188</td>
</tr>
<tr>
<td>Partnership Act (SA), section 24</td>
<td>187</td>
</tr>
<tr>
<td>Partnership Act (SA), section 3</td>
<td>186</td>
</tr>
<tr>
<td>Partnership Act (SA), section 46</td>
<td>185</td>
</tr>
<tr>
<td>Partnership Act (SA), section 9</td>
<td>187</td>
</tr>
</tbody>
</table>
List of Figures

Figure 1: Software Engineering Capability .............................................................. 7
Figure 2: Systems Engineering Process Areas ............................................................ 8
Figure 3: The “Waterfall” Software Life-cycle Model .................................................. 10
Figure 4: The “Waterfall” Software Life-cycle Model .................................................. 11
Figure 5: Percentage Re-work of Software Products for Each Life-cycle Phase .............. 12
Figure 6: “Spiral” Software Development Life-cycle Model ........................................ 12
Figure 7: “Spiral” Lifecycle Model Schedule Milestones ............................................ 13
Figure 8: Monthly Rate of Requirements Change for Selected Projects ....................... 17
Figure 9: Monthly Rate of Requirements Change for Selected Projects ....................... 17
Figure 10: Cost of Change V Software Life-cycle Phase ............................................. 18
Figure 11: Software Defects ......................................................................................... 26
Figure 12: Link the Contract to Software Engineering “Best Practice” ............................ 85
Figure 13: Software Engineering Capability Maturity Levels .................................... 88
Figure 14: Capability Maturity Levels - Key Practices ............................................... 88
Figure 16: Case Study 1 - Summary of Claims ......................................................... 99
Figure 17: Case Study 2 - Summary of Claims ......................................................... 122
Figure 19: Cost of Change V Software Life-cycle Phase ............................................. 130
Figure 20: Case Study 3 - Summary of Claims ......................................................... 131
Figure 22: Case Study 4 - Summary of Claims ......................................................... 144
Figure 24: Case Study 5 - Summary of Claims ......................................................... 157
Figure 26: Outline of [Proposed] “Information Technology Systems Integration and
Outsourcing Code of Conduct” ....................................................................................... 172
Figure 27: Categories of Trust ...................................................................................... 192
Figure 28: Relationship Models: Method of Analysis ................................................... 202
Figure 29: Relationship Model Evaluation Criteria ..................................................... 203
Figure 31: Primary Risk Categories ........................................................................... 214
Figure 32: Evaluation of Conventional Model Attributes .......................................... 220
Figure 34: Evaluation of Service Model Attributes ..................................................... 232
Figure 36: Evaluation of Captive Model Attributes .................................................... 244
Figure 38: Evaluation of Joint Venture Change Agent Model Attributes ...................... 258
Figure 39: Comparison of Relationship Models ......................................................... 264
Figure 40: Full List of Relationship Model Findings .................................................... 270
Figure 41: Micro-agreements – Linking the Disciplines .............................................. 276
Figure 42: Where Do Micro-agreements Get Their Force? ........................................... 279
1. Introduction and Objectives

Information technology plays an increasingly important role in the daily lives of most people one way or another. The development of large information technology systems however historically tend to have a real risk of cost and schedule blowouts, poor quality or even total failure. This document aims to address particular aspects of modern large information technology system acquisition practices with the aim of identifying specific legal attributes of such projects. It also aims to address how those attributes work in conjunction with modern information technology project management practices.

During the course of a large information development, communication between the parties is paramount to ensuring that all components fit together to form a system capable of meeting customer requirements. In the course of development it is usual for engineers and software developers to communicate extensively both orally and in writing to address specific technical issues. It is also likely that at the technical level, many “micro-agreements” are made covering such aspects as preferred options for system design and implementation; data formats etc. The focus of this work is on these micro-agreements. Thus a “micro-agreement” is an agreement or common understanding between two or more parties involved in an information technology or related development or acquisition project, which relates to some aspect of work associated with that project, but is not necessarily expressly or by implication covered by any existing or planned contractual clauses.

The aim is to examine these micro-agreements in the context of large information technology projects to obtain a better understanding of how they effect legal rights and obligations, and how they can be managed effectively from the legal point of view.

Thus in summary:

The Objectives of this document are:

- to discuss the general nature of large information technology projects and highlight any common obstacles to success from an engineering and project management perspective
- to discuss the law applicable to information technology contracting (including a number of related legal concepts such as estoppel) which affect legal rights and obligations
- to introduce the new concept of a “micro-agreement” and identify the benefits and disadvantages of them through a number of case studies specific to the information technology industry;
- to examine the legal relationships which may be present in larger information technology developments and to assess the relative impact of micro-agreements on each of these relationship models, and
• to make specific recommendations to allow the benefits of micro-agreements to be maximised and the adverse effects to be minimised.

The objectives are intended to encapsulate research conducted on the new concept of “micro-agreement” which was conceived in this work. Micro-agreements are examined in the context of two primary disciplines – law and information technology. An attempt has been made to consider practical issues raised in a number of case studies and to list specific findings and recommendations for the effective use and management of micro-agreements.
PLEASE NOTE

The greatest amount of care has been taken while scanning the following pages. The best possible results have been obtained.
2. The Nature of Large Information Technology Projects

The objective of this chapter is:
- to discuss the general nature of large information technology projects and highlight any common obstacles to success from an engineering and project management perspective

2.1 INTRODUCTION

In order to achieve the overall objectives of this work, as a starting point, this chapter provides some insight into the software development process, and other factors that are important to large-scale developments. It also aims to provide some quantitative data on the size, complexity, defect rates, and other factors including the rate of change which can usually be expected in such projects. Overall, it is intended that this chapter will give a profile of large information technology systems integration contracts to identify attributes that are peculiar to such projects. It is not intended at this stage to provide any legal analysis of such developments but merely to provide an engineering and business perspective.

Once a reasonably clear view is established of the technical and "business" environment within which large software developments take place, later chapters will then concentrate on the legal issues associated with such developments.

2.2 BACKGROUND

2.2.1 Introduction

Large software developments can be difficult and costly to produce for a number of reasons. One of the most difficult factors which developers have to deal with is the uncertainty in estimating the size and cost of projects to begin with. Without clear and accurate estimates as a starting point, any contracting process that relies on assumptions is likely to cause problems. In addition, software development projects often are conducted in an environment where a significant rate of change exists, not only in "end products" such as the source-code\(^1\), but also the fundamental requirements of the system.

\(^1\) "Source-code" is a term used to denote a human readable form of a computer program which, once written, can be then translated into a form which is machine readable (i.e. "Object-code").
As will be seen, constant change is a characteristic of large software projects which, for any successful developments, contracts must take into account.

The geographical nature and relative isolation of Australia has in the past forced Australians to use a significant amount of ingenuity and enterprise in overcoming problems associated with large engineering developments. Despite these difficulties however, many large engineering projects have been successfully completed. Changes to Australian government policies over the past few years have encouraged the undertaking in Australia of large engineering projects in a variety of disciplines including telecommunications, defence electronics and general computer applications\(^2\). These projects provide opportunities for Australian industry to develop the skills and infrastructure to not only be relatively self-reliant in many engineering domains, but also to compete internationally in high technology industries. By necessity however these projects rely on Australian and international laws to provide a degree of certainty to the parties, and to provide the framework for establishing the rights and responsibilities of those involved.

The rate of technological advance is currently so high that advances in specific applications are often redundant within a few years due to emerging technologies. Accordingly, high pressure is placed on the developers and manufacturers of such products to continually reduce development periods and costs while at the same time maintaining the quality of their products and services. In the internationally competitive environment of high technology engineering it is imperative that as few constraints as possible are imposed on developers, manufacturers and associated industries. In order for information technology developers to build and maintain a high technology industry, as much support as possible needs to be provided by the legal system to allow the solution of complex engineering problems to be solved efficiently.

As an example of issues that arise in large projects, it is quite possible that at the time of signing the initial contract a significant part of the detailed requirements are undefined. In contrast, a contract for the development of a small project usually includes a specification which can define reasonably completely the functionality, performance and interfaces of the final product. Moreover, in the larger projects problems are often compounded by the likelihood of many parties being involved on a sub-contractual or other basis. It is thus conceivable that the amount of contractual certainty may be low and that conversely, the risks high. Naturally effective project management and engineering practices will mitigate these risks to some degree although the contractual framework will undoubtedly have a major influence on the success or otherwise of the engineering endeavours.

### 2.2.2 How Large is Large?

Overall, the intention of this research is to concentrate on larger information technology system developments including systems integration contracts. These contracts are usually associated with software projects that can be classed as large. But how large is large?

Traditionally, "large" means big and in the context of computers, the ENIAC computer was physically very large. That machine took up a room and consisted of a "large U-shaped assemblage of 40 panels which together contain approximately 18,000 vacuum tubes and 1500 relays"\(^3\). Despite this, large in the present context has a slightly different meaning.

---

\(^2\) For example, the Commonwealth Department of Defence established the "Commercial Support Program" to encourage Australian industry to participate where possible in Defence support activities. Defence Circular Memorandum 91.

"There is 750Kb of code in the newest Ford transmission system, 500Kb in a Phillips TV set." These computer programs would be considered very large if they were developed even 10 years ago, however by today's standards, they could only be classed as less than average size. The term "large" therefore can only really have meaning if it is used in context as a relative rather than absolute measure. Accordingly, mere code size alone is perhaps insufficient to categorise a "large" information technology project.

Examples of engineering "mega-projects" which contain a large information technology component and which are currently either under development or have been recently completed include the Fibre Optic Link Around the Globe (FLAG); the Twin Towers of the Kuala Lumpur City Centre in Malaysia; Hong Kong's Chek Lap Kok Airport; the Boeing 777 jetliner; and the International Space Station. All of these projects are among the largest of their type ever undertaken. Each includes large amounts of computer technology both in the design and implementation stages, and all are based on multinational partnerships of many companies. The Boeing 777 airliner, for example contains over 2.6 million lines of code incorporated in the avionics and cabin entertainment system. Twenty percent of the aircraft was supplied by almost 60 international companies including (some) Australian ones.

In Australia, an example of a large project is the Collins Class submarine. To put the size of the project in perspective, it has been described in the following terms:

"It's a massive software integration program - bigger than anything which has been done, anything which is around in other ships, bigger than has been done in major components of the space program - including even putting men on the moon." 

The project consisted of an engineering development that spanned approximately ten years, engaged many hundreds of engineers and scientists over that time, and had a budget more than five billion Australian dollars. The software size has of this project has been estimated at approaching 1000 person-years of effort. To a large extent, the project has been successful although criticism has been raised in relation to a number of issues. These issues include the schedule where there was a (greater than) 18 month delay in delivery; cost overruns (which in this case were mainly absorbed by the prime contractor); and software quality (where upon delivery of the first submarine the highly computerised combat system "carried a list of bugs running to several pages").

Despite attempts by the Department of Defence to put the problems into perspective, the Australian National Audit Office found that problems did exist within that and other Australian Defence projects. In summary, the Audit Office found that of sixteen projects examined (including the Collins submarine project), the most common problems were:

---

4 Watts Humphrey, as cited in Sydney Morning Herald article "One Mans Quest for Perfection", 3 September 1996 p3c.


6 Professor Des Ball, Strategic & Defence Studies Centre, Australian National University, during discussion on the Australian Broadcasting Commission television programme: Four Corners: Deep Trouble, being a story broadcast on Monday 24 May 1999.

7 Blair, Colin: "Collins' Platform is Fine but Combat System still needs Tuning", Asia-Pacific Defence Reporter, September-October 1996 page 27


9 "It might have a smell of truth about it, but that tends to be exaggerated out of all proportion. ...", being comments by Commodore Eoin Asker as reported in: The Age, 7 February 1999.


Paul P. Parnell
• "inadequate evaluation of project proposals;  
• under-estimates of project costs, time scales and risks;  
• incomplete project planning;  
• inadequate evaluation of tenders;  
• contracts which did not specify all contract requirements or provide effective incentives for contractors to minimise cost or perform to schedule;  
• inadequate monitoring of contractor performance;  
• contract supervision which did not submit the (sometimes) large number of contract changes to sufficient scrutiny to preserve project budgets and schedules;  
• and generally slow Departmental decision making processes".

The following Sections and chapters of this document aim to focus more clearly on the problems associated with large scale software developments, to analyse the problems by introducing the concept of "micro-agreements", and to recommend strategies and methods for overcoming at least some of these problems. Stated in another way, the list of problems identified above by the Australian National Audit Office includes aspects related to requirements management, project planning, sub-contractor management, quality assurance, and project tracking and oversight. These specific areas are the subject of case studies provided in Chapter 4: "Micro-Agreements" Part 1: Concepts and Case Studies". In that chapter, some of the legal, technical and managerial issues are examined in the light of each of these areas and a number of recommendations are made.

In summary, traditionally large information technology projects have been delivered late, have cost more and have had significant quality problems. This issue is discussed in more detail in the following Sections although for the present purposes it is the larger "more difficult" information technology projects that this document is aimed at. Although a number of examples of large projects have been cited above, for the purposes of this work therefore it is considered in general terms that a large systems integration contract is more likely than not to involve information technology projects:

a) that have a time-span of perhaps many years;
b) that have significant elements of research and development, or at least require innovative solutions, or which can be considered technologically complex;
c) where the work may be conducted by many organisations and individuals;
d) where parts of which the work may be conducted in different geographical locations, including internationally; and
e) where the costs may be large.

The above comments have been made in order to set the context within which to identify specific problems associated with large information technology projects. The following Sections therefore aim to provide more specific examples of issues that arise in large projects, particularly from the point of view of information technology and systems engineering.

2.3 SYSTEMS ENGINEERING

---

12 see for example: Glass, R. Software Runaways, Prentice Hall, 1998; Flowers, S. Software Failure: Management Failure: Amazing Stories and Cautionary Tales, John Wiley & Sons, 1996; and Bohem, B. "Making RAD Work for your Project", IEEE Computer, Vol. 32 No. 3, March 1999 p.113; also see the discussion above in relation to the COLLINS Submarine Project.
2.3.1 Introduction
This Section provides a general discussion of some aspects of systems engineering and is not intended to be an exhaustive coverage. Further Sections isolate and discuss specific areas in more detail, with the aim of ultimately identifying issues that should be taken into account when writing and managing systems integration contracts.

2.3.2 A Definition
The formalities of large software development projects are often embodied in the discipline called “Systems engineering”. This has been defined as:

“A logical process of activities which transforms a set of requirements arising from a specific mission objective into a full description of a system which fulfils the objective in an optimum way. It ensures that all aspects of a project have been considered and integrated into a consistent whole.”\(^{13}\)

In relation to large software development and integration projects, the above definition includes a number of key factors that are relevant to the way in which the software development business operates. Specifically, the definition refers to a process, that is, “a systematic series of actions directed to some end”\(^{14}\). Software Process Engineering and development capability is relevant to the way in which software developments occur and is an emerging trend within the large scale software development field. In itself it is a broad topic although in summary, it is concerned with formally addressing a number of factors such as organisational and resource management, the software engineering process, and tools and technology\(^{15}\). The following diagram gives a basic pictorial outline of the primary issues normally addressed in a software engineering context.

![Diagram of Software Engineering Capability]

In the diagram above, all the primary issues of Software Engineering capability are outlined. These primarily include:

- Organisational Structure
- Resources, Personnel, and Training
- Technology Management
- Documented Standards and Procedures
- Process Metrics
- Data Management and Analysis
- Process Control


\(^{14}\)The Macquarie Dictionary, 1981

\(^{15}\)Software Engineering Institute: Key Practices of the Capability Maturity Model, 1991
The above diagram represents the state-of-the-art software engineering process model however strictly speaking, at the systems engineering level, other factors also need to be considered. A system may consist of not only software but also hardware, logistic support (for example spares, training etc) as well as many other factors. Specifically, the Systems Engineering Capability Maturity Model\(^{16}\) addresses the major elements of a full systems engineering capability by identifying requirements in the following key process areas:

<table>
<thead>
<tr>
<th>Process Area</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA01</td>
<td>Analyse candidate solutions</td>
</tr>
<tr>
<td>PA02</td>
<td>Derive and allocate requirements</td>
</tr>
<tr>
<td>PA03</td>
<td>Evolve systems architecture</td>
</tr>
<tr>
<td>PA04</td>
<td>Integrate disciplines</td>
</tr>
<tr>
<td>PA05</td>
<td>Integrate system</td>
</tr>
<tr>
<td>PA06</td>
<td>Understand customer needs and expectations</td>
</tr>
<tr>
<td>PA07</td>
<td>Verify and validate system</td>
</tr>
<tr>
<td>PA08</td>
<td>Ensure quality</td>
</tr>
<tr>
<td>PA09</td>
<td>Manage configurations</td>
</tr>
<tr>
<td>PA10</td>
<td>Manage risk</td>
</tr>
<tr>
<td>PA11</td>
<td>Monitor and control technical effort</td>
</tr>
<tr>
<td>PA12</td>
<td>Plan technical effort</td>
</tr>
<tr>
<td>PA13</td>
<td>Define organisation’s systems engineering process</td>
</tr>
<tr>
<td>PA14</td>
<td>Improve organisation’s systems engineering process</td>
</tr>
<tr>
<td>PA15</td>
<td>Manage product line evolution</td>
</tr>
<tr>
<td>PA16</td>
<td>Manage systems engineering support environment</td>
</tr>
<tr>
<td>PA17</td>
<td>Provide ongoing skills and knowledge</td>
</tr>
<tr>
<td>PA18</td>
<td>Co-ordinate with suppliers</td>
</tr>
</tbody>
</table>

**Figure 2: Systems Engineering Process Areas**

Research results exist which show the benefit achievable through the introduction and use of a solid engineering process for the development of large information technology projects\(^{17}\). In one study\(^{16}\) for example, results of process factors showed that the following benefits were achievable:

- Defect reduction of up to 90%,
- Productivity gains of up to 350%, and
- Schedule reduction of up to 70%

Of course the above figures are “best case” however serve to illustrate the potential power that a solid engineering process can play in the success of a project.

---


---

Paul P. Parnell
Although the above models are detailed and encompass many issues, for the purposes of this work, the primary focus will be on the software engineering process that can be used to develop large information technology products. Specifically, topics such as the software life cycle, requirements management, metrics and measurement, and defects will be addressed.

2.3.3 Engineering “Trade-offs”

Traditionally, the engineering discipline has been associated with trade-offs (that is, concessions made in order to satisfy a number of competing needs) and the software and information technology fields are no exception. As in most engineering disciplines more than one possible implementation of a particular design may exist. Trade-offs generally need to be made in the areas of cost, quality and schedule in order to satisfy customers' requirements and expectations.

In building a bridge for example, engineers are faced with a multitude of different possible designs that will result in structures having different physical properties, different costs and different construction time-scales.

As with a bridge designer, a computer engineer is faced with a number of choices when developing a system. Depending upon the customers' requirements, a particular system may be implemented in a variety of ways. In some instances, certain functionality and performance may be achieved by using either hardware or software design techniques, or a combination of both. For example, with the introduction of gate array technology it is becoming more common for electronic hardware engineers to develop circuitry by “programming” general-purpose devices\(^ \text{19} \) to perform particular functions. Whether the engineer implements a circuit using traditional circuit design methods or 'programs' a gate array does not generally alter the functionality or performance of the device (although circuit board space or power requirements may be affected by the decision).

Overall, from the customer point of view it is usually the case that the particular design does not matter so much provided that the overall customer requirements are met. Taking the above examples, provided the bridge can carry the required volume of traffic between the required points, it is likely that the actual design will not be of major concern to the purchaser (although naturally in some cases, aesthetics and other factors will be a consideration).

From a software development or systems integration perspective, the point is therefore that if the software developers are not free to implement particular requirements in one of a variety of ways, then it is more likely that designs will be “locked in” to possibly inefficient or inappropriate implementations early in the development cycle.

System contracts therefore ideally should allow scope for engineering trade-offs to be made throughout the life of the project and for trade-off studies to be undertaken as appropriate. Flexibility is the key to allowing changes to be made throughout the life of a project although of equal importance is ensuring that these changes are managed effectively.

2.4 THE SOFTWARE LIFECYCLE

2.4.1 Introduction

\(^{19}\) such as Gate Arrays or Electrically Programmable Logic Devices (EPLD)
When commencing work on a significant software development of integration project, engineers traditionally break down the task into a series of manageable segments. Over the years a number of different models have been developed which aid in the planning and tracking process, although in general all of these life-cycle models identify work including:

a) determining what the system and software is required to do;
b) determine how it is to do it;
c) implement (code) the software; and
d) test and integrate the software.

### 2.4.2 Lifecycle Models

A number of software development life-cycle models have been developed however for the purposes of this chapter two will be considered - the Waterfall model, and an iterative model.

#### 2.4.2.1 The Waterfall Model of Software Development

The classical "waterfall" life-cycle model incorporates the activities listed above, together with others, into a number of clearly defined steps as shown in the diagram below. Specifically, the following phases are identified: Requirements Analysis, Design (Preliminary and Detailed), Coding, Testing and Integration.

![Figure 3: The “Waterfall” Software Life-cycle Model](image)

This waterfall model has been in existence for a number of years and is embodied in some older software development standards such as DOD-STD-1679, and DOD-STD-2167. One thing to bear in mind however is that the model is somewhat overly simplistic and does not entirely represent the way in which larger software developments are actually conducted.\(^\text{20}\)

One of the fundamental aspects of the classical waterfall model as described above is that each project phase is completed before the next phase is commenced. At each step in the life cycle, specific documents and other products are produced by the developer and

---


Paul P. Parnell

10
accepted by the purchaser before formal commencement of the next phase. The idea behind this model is quite simple: at the end of the Requirements Analysis Phase a total, complete specification of what the system is required to do is (in theory) available. Similarly, for each of the other phases of the project, formal and complete documentation is available for use in the subsequent phase.

Using the waterfall model, the optimal point at which a contract should come into existence is at the completion of the requirements analysis phase, when (in theory) a complete and accurate description of the system or software to be eventually delivered, is available. In practice, most systems integration projects that involve the development of software of any significant size, require contracts to be signed at the beginning of the process, even before the requirements analysis phase has been undertaken.

It is not intended in this Section to examine the possible contracting models but merely to point out that at the time of the signing of a major software development contract, a significant part of the requirements would not yet have been defined. This premise is based on a fairly simplistic software life-cycle model, the Waterfall Model, which in an ideal world have stable products at the end of each phase.

2.4.2.2 Iterative and Evolutionary Software Development Models

In the real world however things are not quite as straightforward as the classical Waterfall Model would predict. The main problem with the Waterfall Model is that, despite attempting to clearly isolate each of the development phases and the products of each of those phases, in practice a certain amount of re-work is required of products produced in each of the phases, as a result of work performed in subsequent phases. In essence, the software evolves over time as a result of new information that becomes known as time goes on. Diagrammatically, this could be represented by the Waterfall Model with suitable re-work or feedback mechanisms such as shown in the following diagram:

![Figure 4 - The "Waterfall" Software Life-cycle Model with Feedback](image)

The above diagram shows that information derived in subsequent software life-cycle phases may be used in earlier phases by way of "feed-back" loops. How much re-work can be expected? The following table provides empirical data identifying that, for one company’s historical analysis of software projects, re-work on each phase can be up to 25%.


Paul P. Pamell
<table>
<thead>
<tr>
<th>Development Phase</th>
<th>Re-work Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements Analysis</td>
<td>25%</td>
</tr>
<tr>
<td>Preliminary Design</td>
<td>25%</td>
</tr>
<tr>
<td>Detailed Design</td>
<td>25%</td>
</tr>
<tr>
<td>Coding</td>
<td>10%</td>
</tr>
<tr>
<td>Testing</td>
<td>10%</td>
</tr>
<tr>
<td>Integration and System Testing</td>
<td>25%</td>
</tr>
</tbody>
</table>

**Figure 5:** Percentage Re-work of Software Products for Each Life-cycle Phase

Sometimes this rework is incorporated into other software life-cycle development models (apart from the Waterfall Model). For example, the development can be undertaken in a way which work in a number of phases can be undertaken simultaneously, but at the same time being fed-back to the products of ‘earlier’ phases. Later software development standards such as DOD-STD-2167A, MIL-STD-498 and IEEE/IEA 12207 allow more readily for this type of development model. The following diagram shows one representation of a “spiral” model\(^{22}\) that attempts to incorporate such issues.

![Requirements Analysis → Design → Coding → Testing → Integration](image)

**Figure 6:** “Spiral” Software Development Life-cycle Model

Another way of representing the “spiral” model is shown in the following schedule diagram. This represents an “unwound” spiral with the tasks scheduled against time\(^{23}\).

This figure shows the practical scheduling of major activities conducted for a software project using the spiral approach. In summary, a series of prototypes are developed and risk assessments performed amongst the more traditional requirements, design, coding, test and integration work.

---


\(^{23}\) This schedule was derived from a spiral model diagram provided in: McConnell, S. Rapid Development – Taming Wild Software Schedules, Microsoft Press, 1996

Paul P. Parnell 12
Quantitative studies\(^{24}\) have shown that the practices of prototyping, less rework of software products, and employing better programmers are the three things most likely to result in a reduced schedule. The spiral model has direct support for the first two of these practices. Specifically, it can be seen that prototype software products are available early in the lifecycle and the reviews conducted help to establish firmly the requirements and architecture of the system early. Potentially rework is reduced because a clear understanding and definition of the requirements of the system is available at an early stage.

Overall, the above discussion has focused on a few of the life-cycle models that have been used in the industry. Some models may be more applicable than others and comparisons are available in the literature\(^{25}\).

### 2.4.3 Change

The lifecycle models discussed above provide some examples of how the logical steps associated with software development can be structured. Despite this logical structuring of work, it must be understood that, in a real-world day-to-day development, things change. Contracts that assume that systems requirements, design or any other products of the development effort are static are likely to be unrealistic and may cause more problems than they prevent. Experience has shown that constant change is a reality of major information systems developments. Systems engineers in the past have developed a number of formal approaches to ensuring that change is minimised, and these

---


\(^{25}\) For a comparison of the strengths and weaknesses of each model, see: McConnell, S: Rapid Development - Taming Wild Software Schedules, Microsoft Press, 1986
approaches include software development standards, requirements analysis methodologies, and software process assessment schemes.

Williams compares software with an "organism" which evolves.

"Evolution is a useful analogy for discussing software development, because software more closely resembles an organism than a mechanism. It's rare that software is simply manufactured and used as is forever. Just as a living organism adapts to its changing environment, software must adapt to ever-changing requirements. This is as true during development, when the parts to be integrated and tested also change from day to day, as during the rest of a system's life cycle. An evolutionary approach to software development can accommodate changes as they occur. This approach acknowledges that you don't know all the details ahead of time and that those that do exist may change."

Overall it is generally recognised that coping with constant change is an essential part of the way in which information technology developments occur. This is true from both the technical perspective as well as from the legal or contracting perspective. The consensus in today's software engineering world is that a dynamic iterative approach (rather than a static approach) to development generally achieves better results.

2.5 REQUIREMENTS MANAGEMENT

2.5.1 What are Requirements?
Requirements are "not things 'out there' flying about like butterflies". Classically, the requirements constitute a complete statement of what the system will do without referring to how it will do it. Although there has been some criticism of this view as being overly simplistic, it is a concise way of defining the term that is suitable for the present purposes.

In practice, software engineering related standards tend to separate requirements into three primary areas: capability (or functional) requirements, performance requirements, and interface requirements. Ideally, the specification of these three primary areas should describe what the system is required to do.

2.5.2 The Problem
Previous Sections identified a number of phases that are generally needed to complete a large information technology project. The first substantial phase - Requirements Analysis - is intended to clarify "what" the system is required to do. The end product from this phase is ideally a complete specification defining the functionality, performance, and interfaces required in the final system, and in summary:

"...The functional requirements specification is complete when the project team is willing to commit itself to a set of functions, recognising that the completeness of the document

---

31 For example, ISO/IEC 12207 Information Technology - Software Life-cycle Processes, 1995-08-01
32 or more recently referred to as "requirements engineering"
describing them is vital to the quality of the final system, and that any changes made to the
document after it is released to the design engineers will directly affect product quality,
production schedule, and life-cycle costs.\textsuperscript{33}

The problem is however, that requirements usually continue to change even after the
Requirements Analysis has been conducted and the Functional Requirements
Specification is delivered:

"For a long time the largest source of project delays, redesign typically becomes necessary
because the product requirements were poorly defined or changed, the engineering
specifications were inferior, or the basic design's ability to meet the original requirements
were not thoroughly reviewed. The full effect of inadequate planning is rarely detected until
late in the development cycle, sometimes even into the system's design validation phase.
The later any design defects are detected, the more time and money redesign will need.
Moreover, late changes often compromise the products reliability and maintenance costs,
perhaps even affecting its profitability and success. Spending extra time up front
completing and clarifying the basic requirements always pays high dividends in the end."\textsuperscript{34}

Although the above quotation identifies that incomplete requirements specifications are
the cause of many problems associated with information technology developments, what
it does not explain is that, as previously mentioned, change is a fact of life in such projects
and accordingly effective requirements management processes should ideally be in place
to ensure that the impact of change is minimised.

Why is it such a problem? From the contracting point of view, ideally the parties to a
contract need to operate in an environment that provides some level of certainty and
predictability. To properly assess the risks associated with any development, the usual
approach is to include as part of the contract or tender, a Statement of Requirement, or
other document which identifies a complete set of user requirements. Ideally, this
document should include information on all functionality, performance and interfaces to be
delivered in the final system. Invariably, due to business pressures such as short time-
frames for development, information technology contracts quite often come into being at a
time when the full requirements are not yet known.

In other words - and this is an important point - after the contract is signed, it is more likely
than not that requirements will change.

2.5.3 Why Do Requirements Change?

One useful explanation of the reasons why software requirements change is provided in
the following statement:

"Requirements change because software is changing the ways companies operate. In a
sense, creating software requirements is like hiking in a gradually lifting fog. At first only the
surroundings within a few feet of the path are visible, but as the fog lifts, more and more of
the terrain can be seen. ... Not all software applications have unstable requirements but the
majority do. Certain kinds of engineering and scientific software, and software that is
embedded in physical devices (such as automotive fuel injection systems), may reach a
point of stability early. In fact, once these requirements stabilise, they may stay constant for
several years. However when software deals with business factors, change is almost
inevitable."\textsuperscript{35}

\textsuperscript{34}Skytte, K. Engineering a Small System, op cit, p. 63
\textsuperscript{35}Jones, Capers. Strategies for Managing Requirements Creep, I.E.E.E. Computer, Vol. 29, No. 6, June 1996,
p.92
As an example of the business factors mentioned above, in a competitive environment it is likely that any product under development will need to take heed of functionality and features present in competing products, and provide at least equivalent or enhanced functionality. This continual "leap-frogging" of requirements can mean that software developers are continually forced to chase a moving target. In addition, once the initial prototypes are developed and users start to use the system, often additional needs become known. The relatively common practice of "alpha" and "beta" testing (that is - testing at selected sites prior to formal release into the marketplace) is designed to not only show up "bugs" (defects), but also to give the users an opportunity to use the system in real-life situations and identify further requirements as necessary.

In order to attempt to state the problem more specifically, the following observation is relevant:

“One of the most chronic problems in software development is the fact that application requirements are almost never stable and fixed. Frequent changes in requirements are not always caused by capricious clients (although sometimes they are). The root cause of requirements volatility is that many applications are attempting to automate domains that are only partly understood.”

Although this statement may be true, it is submitted that it is not the only reason. The technology of computing hardware is in some ways allowing new applications to be developed that in the past may have been prevented because of a lack of computing power. The rate of change in hardware capability is often summarised by Moore's law. That "law" states that processor power and memory size double per unit cost every eighteen months (approximately). This rapid rate of change gives software developers the platform for developing new and improved products that may not have been possible only a few years earlier. Any software product currently under development will need to take into account not only the actual computing power available at the time of development, but also newer technologies that may be available later.

**2.5.4 How much do Requirements Change?**

Regardless of the reasons why software requirements change, by how much do they change? In any field of human endeavour some change can be expected, however to get an idea of the typical amount of change that takes place over the life of an information technology development or integration project, quantitative studies have been conducted. The following two tables summarise data collected which provides an indication of the likely amount of change which will occur in requirements firstly, per month; and secondly, over the life of the project.

<table>
<thead>
<tr>
<th>Software Type</th>
<th>Monthly Rate of Requirements Change to Function Point Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contract or outsource software</td>
<td>1.0%</td>
</tr>
<tr>
<td>Information Systems Software</td>
<td>1.5%</td>
</tr>
<tr>
<td>Systems Software</td>
<td>2.0%</td>
</tr>
<tr>
<td>Military Software</td>
<td>2.0%</td>
</tr>
<tr>
<td>Commercial Software</td>
<td>3.5%</td>
</tr>
</tbody>
</table>

38 Moore's law is not a legal concept, but is rather a mathematical observation of measured computing power and memory size versus cost which has been used to accurately predict technology improvements for the past two decades.
Figure 8: Monthly Rate of Requirements Change for Selected Projects

<table>
<thead>
<tr>
<th>Software Type</th>
<th>Requirements Change to Total Function Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contract or outsource software</td>
<td>14.45%</td>
</tr>
<tr>
<td>Information Systems Software</td>
<td>17.95%</td>
</tr>
<tr>
<td>Systems Software</td>
<td>21.50%</td>
</tr>
<tr>
<td>Military Software</td>
<td>25.54%</td>
</tr>
<tr>
<td>Commercial Software</td>
<td>19.75%</td>
</tr>
</tbody>
</table>

Figure 9: Monthly Rate of Requirements Change for Selected Projects

The above data was collected using the technique of Function Point Analysis which essentially allows for measurements to be made of the size of software by taking into account software attributes such as: inputs, outputs, inquires, logical files and interfaces. To derive the above data, project size was measured at the completion of the requirements analysis phase (when the requirements are considered to be relatively firm and stable), and then measurements of the size were taken throughout the life of each of the projects. Comparing the initial and final measurements gives a direct measure of the rate of change of requirements over the life of the project. (Further discussion on function point analysis is provided in the later Section: Metrics and Measurement).

It is generally acknowledged that the Function Point Analysis method used to derive the information in the above tables is not absolutely accurate however for the purposes of this document it is considered sufficient to give a reasonable indication of the rate of change which can be expected in the requirements of a system over its development life-cycle.

What then does the data in the above two tables mean? Depending upon the type of software being developed, that is, information systems software, military software etc., it can be expected that around 15%-25% of requirements will change over the life of the project. The consequence of this change to requirements is that following phases of the software project, (that is, design, coding, test, and integration) will also be affected. Although no empirical data is presented here to show the absolute increase in cost and schedule associated with this increase in requirements, in the authors experience it is considered that such increases in requirements would lead to at least similar levels of increase in both cost and schedule, but probably more.

Invariably, if an information technology contract locks the supplier into a set of requirements (possibly identified in a “Statement of Requirement” attached to a contract) then either the supplier, the purchaser or both will be disadvantaged in some way as cost, schedule, and quality of the product are artificially massaged in an attempt to meet the contract.

The above data shows that even after a formal requirements analysis phase has been conducted, requirements still change significantly. Unfortunately many information technology contracts are put in place even before the requirements analysis phase has been conducted, which almost guarantees that the rate of change of requirements will be high after the contract has been signed. Unless the contract is set up carefully to cope

---


Paul P. Parnell
with this change, the parties to the contract are likely to operate under a highly unstable and changing environment which the contract attempts to control.

In relation to the cost of change, studies have shown that it is usually more expensive to include a new or changed requirement later in the project rather than early on. Specifically, the following graph illustrates this point. Based on the figure below, it would cost over 100 times more to add or change a requirement after it is delivered and is operational, than if that same change was identified during the requirements phase.

![Figure 10: Cost of Change V Software Life-cycle Phase](image)

### 2.5.5 How to Minimise Changes in Requirements.

#### 2.5.5.1 Introduction

There are a number of techniques that the software industry is currently applying to the problem of requirements management, and some of these are identified in following Sections. The aim is not so much prevent new requirements from emerging but to manage these in a constructive way.

> "In Japan, such companies as Matsushita Communication Industrial Co.'s Instrument Division in Yokohama and Toshiba American Medical systems Inc. in Tustin, Calif., manage both to prevent and profit from creeping elegance [i.e. requirements changes]. As new ideas or requirements surface, they are collected not for ongoing projects, but for later upgrades or new products."\(^{41}\)

Although the above method may help to alleviate some of the problems associated with dynamically changing requirements, it does not solve the basic problem. In the latter stages of a project it may be useful to collect new ideas and requirements for later versions of the software however during the early stages it is important to try to fully understand what the system has to do. The usual way of doing this is to ensure a relatively complete and accurate specification is available to start with. More modern approaches such as evolutionary development can adapt to changing requirements by evolving the system over time, as more information becomes known.\(^{42}\)

---


\(^{42}\) "Evolution is a designed characteristic of a system development which involves gradual stepwise change"
From an engineering or software development point of view, what methods and tools are available to help aid in the analysis of requirements? Jones lists a number of methods each of which is described briefly below. From the point of view of information technology contracting, it is important that every effort is made to ensure that requirements are managed appropriately and to that end it may be worthwhile considering that the contract provides for one or more of the following methods to be incorporated in the development process, or at least, that the contract is able to accommodate the ideas embodied in the following mechanisms. Further discussion and case studies on contracting software engineering processes are provided in Chapter 4.

2.5.5.2 Joint Application Development
Joint Application Development is a method whereby both the user and the developer work together to identify the requirements of the system. It is usual that a facilitator is also present to aid both parties in the process of reaching a consensus on the requirements to be implemented. This method was developed in Canada in the 1970’s and is now used effectively by information systems developers. Overall, if implemented correctly, it has been estimated that this method may cut the rate of requirement change by around half throughout the life of the project.

Apart from reducing the amount of requirements change associated with a project, studies have also shown that Joint Application Development can reduce the total development time of a project by around 5-15 percent.

2.5.5.3 Prototyping
A powerful tool for getting user feedback early in the development cycle is the prototype. Essentially a working model of the software is built (probably with reduced functionality) with the aim of allowing the user to gain insight into how the system will eventually work, and provide comment and feedback to the developer on the adequacy of the system. Although users are invariably involved in providing input to the writing of specifications and requirements documents early in the life of the project, it is usual that additional comments and needs are forthcoming if a "hands on" approach is taken. Prototypes facilitate this hands-on approach.

Prototypes generally do not introduce major overhead into the development as they are usually developed as a "one-off". It has been estimated that: "by themselves, prototypes can reduce requirements creep by somewhere between 10 and 25 percent."

The spiral software model has been previously discussed and Figure 7 above provides a simplistic breakdown of the activities associated with that model. Prototyping is heavily used in the spiral development model and accordingly one practical way of ensuring that requirements are managed effectively may be to contractually enforce a spiral model approach to the extent that the activities listed in Figure 7 are represented as contractual milestones.

Glib, Tom: Software Metrics, Winthrop, October 1976, at p.214

Paul P. Parnell
One additional benefit of prototyping is that "when the best solution to a problem is unknown, low-cost, low-risk prototyping can produce considerable results and raise the programmers' confidence so they will tackle more difficult implementations."\(^{47}\)

### 2.5.5.4 Inspections

Failure to use design reviews and code inspections has been cited as one of the worst current development practices.\(^{48}\) Formal inspections can be used to major advantage by systems analysts to ensure that all requirements are captured. The idea behind inspections is essentially the "two heads are better than one" concept. Reviews and inspections by both peers and users can be a simple and effective technique, and some studies have shown that inspections can significantly reduce the rate of change of requirements by as much as 30%.\(^{49}\)

The basic rules of inspections are outlined by Gilb\(^{50}\). In summary, software artefacts such as requirements data, design data, and inspection checklists are provided as inputs to the inspection process. That process includes activities such as planning, preparation, meetings, reworking, and follow-up tasks. The outputs from the inspection process include defect lists, revised requirements specifications and statistical reports. Practical guides to inspections have been published in the literature.\(^{51}\)

### 2.5.5.5 Cost-per-Function-Point Contracts

Function Points can be used as a measure of software size and effort and, accordingly, may be used in contracts to provide some level of certainty:

"Several outsource vendors are exploring the use of cost-per-function-point contracts. This approach allows vendors to use a sliding scale, so the cost per function point rises for late requirements changes. This approach is too new and too experimental to judge the overall effectiveness against requirements creep, but the preliminary results are favourable."\(^{52}\)

Further discussion on Function Point Analysis is provided in the later Section Metrics and Measurement, however for the purposes of the present discussion, the point to note is that software metrics are beginning to be used to measure and estimate software attributes to the extent that such measures can be directly incorporated into contracts. In order to do this however the advantages and disadvantages of function point analysis need to be understood.\(^{53}\) Examples of the use of function points to estimate software effort (and hence cost) can also be found in the literature.\(^{54}\)

### 2.5.5.6 Change-Control Boards

A change control board usually consists of a group of people representing a number of parties who have an interest in the software or system under consideration. It is the task of this board to address all requirements changes and prioritise them with an aim to determining which changes will be implemented in the system.


\(^{50}\) Gilb, Tom: Principles of Software Engineering Management, Addison-Wesley, 1988


Change control boards are primarily used for large systems, particularly in the systems software and military software domains. They are generally effective at ensuring that all prospective changes are formally addressed and, depending upon circumstances including development time-frames, cost and user needs, may decide to either incorporate the change, reject the change, or hold the change over until either its priority is reassessed or other circumstances allow for its incorporation.

Although insufficient empirical data exists to confirm the absolute advantage of this method, estimates have been made which indicate that it can be effective in reducing the rate of requirements changes by around 25%.55

In terms of process improvement activities, some corporations have included this approach as one of the top four priorities56.

2.5.6 Is Requirements Engineering Relevant?

Traditionally a software development contract includes a statement of requirements that defines the system to be delivered. Some commentators have suggested that this model of contracting, that is, the "requirements-as-contract model", is not relevant to software developers today57. Modern software tends to make greater use of existing software components as well as tools and techniques for maximising productivity and accordingly does not require as much requirements engineering as in the past.

"Trends in the last decade - system downsizing, shorter product cycles, the increasing emphasis on building reusable components and software architectural families, and the use of off-the-shelf or outsourced software have significantly reduced the percentage of systems that fit [the profile of a large one off development]" 58.

Despite this point it is generally acknowledged that requirements engineering has a major influence of the final outcome of large software systems. Modern software development standards59 and process models60 explicitly include requirements management activities. Whether the requirements are formalised or not61 it is considered that, particularly for large projects, requirements management is an integral part of the process62.

In a practical sense however the following issues are also important in the modern day context: supporting market-driven inventors, prioritising requirements, coping with incompleteness, integrating design artefacts, making requirements methods and tools more accessible63.

60 Software Engineering Institute: A Systems Engineering Capability Maturity Model, Version 1.1, SECMM-95-01;CMU/SEI-95-MM-003, November 1995

Paul P. Parnell
2.5.7 Conclusion

Overall, the above discussion has attempted to show that change is a fact of life in software development and accordingly any contracting activities need to take it into account. A number of methods are available to ensure that certainty at the time of contracting is maximised and where possible, contracts need to ensure that these and other methods are either provided for in the contract, or at least are not ruled out.

In conclusion, it has been said that:

"Managing requirements changes is essential, because "creeping requirements" affect all downstream deliverables. Moreover, requirements changes become progressively more problematical after the requirements phase has been completed. During the design phase, the average rate of requirements change can exceed 3 percent per month. ... For systems software ... and for commercial software, changes may be due to market needs or pressure from competitive products. The average rate is 2 percent per month from the end of initial requirements until the start of testing. If a competitor suddenly announced a new product, however, monthly change rates could exceed 15 percent."

It needs to be understood that "it is virtually impossible to make all the correct requirements and implementation decisions the first time around". Incomplete or inadequate requirements descriptions are a fact of life in large projects and the legal environment must be set up to allow evolutionary improvements in the software artefacts as new information comes to light.

2.6 METRICS AND MEASUREMENT

2.6.1 Introduction

For a long time the software development industry has operated differently to other scientific and engineering endeavours because of the lack of measurement and predictability:

"The software industry is an embarrassment when it comes to measurement and metrics. Many software managers and practitioners, including tenured academics in software engineering and computer science, seem to know little or nothing about these topics. Many of the measurements found in the software literature are not used with enough precision to replicate the author's findings - a canon of scientific writing in other fields. Several of the most widely used software metrics have proved unworkable, yet they continue to show up in books, encyclopaedias, and refereed journals. So long as these invalid metrics are used carelessly, there can be no true "software engineering", only a kind of amateurish craft that uses rough approximations instead of precise measurement."

Although the above assertions may be a bit harsh they are essentially true although the software measurement field is one of rapid development. Accordingly, significant practical measurement techniques are becoming more common.

In the first place however the question which needs to be asked is: Why measure software?

---


Paul P. Parnell

22
2.6.2 Why Measure Software?

Measurement is generally considered to be one of the cornerstones upon which scientific and engineering endeavours are undertaken. Despite this broad however, a number of issues arise during the conduct of any large scale software development which can be critical to the success or failure of a project. The risks associated with these issues must be managed and one of the primary ways of achieving this is through the use of measurement.

Specifically, the following issues or risk areas have been identified as being common to almost all software development programs.  

- Schedule and progress
- Growth and stability
- Funding and Personnel resources
- Product quality
- Software development performance, and
- Technical adequacy

Measurements can be taken to assess each of the above points, with the aim of tracking them throughout the life of the project to ensure that early warning is given before actual and expected values become significantly different.

From a information technology contracting point of view, metrics and measurement can be important to define, in quantitative terms, a basis for establishing whether certain contract conditions come into play. For example:

"Software outsourcers and contractors can now derive the function-point totals of software during the requirements phase, which means that downstream costs of changing requirements can be quantified. [A] contract might include items like the following:

- Development costs of requirements identified during months 1 to 3 of the requirements phase of the project = $1,500 per function point.
- Development costs for new requirements added during months 4 through 6 = $1,750 per function point
- [etc.]

In the above example, quantitative measurements of the requirements are used directly to cost the contract. Requirements added late in the development are explicitly subject to additional cost commensurate with the additional work required to implement them later in the development life-cycle. Any uncertainty in the rate of change of requirements is minimised as all parties are encouraged to identify requirements during the early stages of the project, while at the same time still have a degree of certainty of cost (etc.) if additional requirements do come to light during later stages in development.

The simple answer as to why we need to measure software is that measurements can help to reduce the risk and uncertainty in a project.


2.6.3 Some Specific Software Measures

There are many measures which can be made to software and its associated life-cycle products although care must be taken to ensure that the correct measure is used for the intended purpose, and that errors in the measurements are clearly known and taken into account. Two specific metrics are discussed briefly below as an example.

2.6.3.1 Complexity Metrics

Software can be complex but how complex? A number of measures are available to answer this question. They essentially measure the number of possible branches that a software program may take. From past experience high levels of complexity in software can lead to high defect levels as well as cause problems during both development and maintenance of the software - if software is complex it can be more difficult to understand and thus more difficult and costly to modify and maintain.

Complexity measurements can thus be used, primarily in the coding stages, to estimate the likelihood of defects being in the system. From a systems integration point of view, they can also be used as a measure of how difficult the integration task will be - integrating a number of software components with high measured complexity is likely to be more difficult than integrating components of lower complexity.

2.6.3.2 Function Point Metrics

Function points have previously been discussed as an example of a particular measure which may be useful in some cases to introduce more certainty into information technology contracts. A good summary of function points is provided by Jones:

"The function point metric, developed within IBM by A.J. Albrecht, has been in the public domain since 1979. It's not perfect, but it is free of the economic distortions of the [Lines of Code] metric.

Function points are the weighted sum of five external attributes of software projects - inputs, outputs, inquiries, logical files, and interfaces - that have been adjusted for complexity. The actual counting rules are quite complex, but they are logical and consistent enough to have been encoded in scores of automatic function-point counting tools. ... The function point total for an application does not change with the programming language. ... Economic costs can be calculated with function points."

Some deficiencies have been identified with the function point metric, although the important point about function points is that, apart from their value as a general measure of software from a technical perspective, they can be used to help define the economics of software by presenting a unit of measure which can be consistently applied to software projects. When purchasing any goods it is invariably essential to know the amount of product being purchased so that reliable estimates of cost, schedules and quality can be built into the contractual equation. Reliable software metrics help to provide some of the same certainty to software developments. In summary,

"As a contractual metric, cost per function point (rather than the primitive [Lines of Code]) helps everyone understand software economics better - clients, accountants, project managers, contract officers, and attorneys. As a matter of fact, the [U.S.] Internal Revenue Service is exploring cost per function point to determine software's taxable value."

69Jones, Capers: Software Metrics: Good, Bad, and Missing, op cit,100
Despite the above comment, newer metrics are starting to emerge which may give the advantages of function point without some of the disadvantages. One of these newer metrics is the "System Meter"71.

**2.6.3.3 Some Areas have no Reliable Metrics**

From the above discussion it can be seen that software metrics can be a useful tool for (among other things) estimating project size and tracking progress. Despite this however a number of areas still remain for which there are no reliable metrics:

"One very important software domain - the data or information associated with software - lacks any metrics whatsoever. As of 1994 there are no known metrics for quantifying or normalising the volume of data contained in a database or repository or used by a corporation. There are no metrics for dealing with the costs of creating, using, or removing data. There is no way to explore the volume of active versus archival data, nor the costs associated with moving data back and forth between active and inactive status. There are no metrics for measuring data quality."72

For these and other areas in which reliable quantitative measures are not available care must be taken to ensure contractual certainty is established by other means. It is not the intention to explore at this point methods to achieve this certainty, but merely to point out that flexibility and an ability to cope with change must be considered in any contract.

**2.6.4 "External" Software Metrics**

As with any type of activities associated with a complex endeavour there are different points of view depending upon the circumstances. The metrics discussed in the preceding Sections can be thought of as "internal" in that they are applied to the software internals itself. Some commentators prefer to look at the "external" metrics73.

"We must stop messing with "internal" software metrics such as complexity and function points and learn really powerful software metrics based on final-product, customer-perceived results such as adaptability, availability, reliability, maintainability, security, portability, and performance. All these quality concepts have shown themselves capable of quantitative expression in practical software-engineering projects."

Despite the above it is important to understand that metrics are the key to developing an effective tracking and oversight process (a specific case study is provide on this point later at Section 4.7). Although the key to effective metrics however is to understand why the metrics are being collected in the first place, and to be clear on what the data will be used for once available.

**2.6.5 Conclusion**

It can be said that measurement is a fundamental cornerstone of science and engineering, although one which unfortunately has generally not been available or used effectively in information technology projects to the extent that it has been used in some other engineering disciplines. The industry has in the past suffered because of ineffective measurement tools however improvements are occurring.

---


---

Paul P. Parnell
Overall, metrics and measurement can be used to introduce greater certainty and reduce risks associated with both the technical and contractual aspects of information technology developments. The key is however understanding why particular attributes are being measured and what the data will be used for once it is available.

2.7 DEFECTS

2.7.1 Introduction

Despite that fact that formal software engineering processes may be adhered to and every effort made to ensure that the developed software is adequate, software defects still occur. Examples of the types of defects which can occur is provided by Taperall:

"publicised examples of the results of computer errors include confused mailing lists, jumbled inventories, scrambled stock quotations, mailings of unaddressed envelopes, incorrect airline reservations, clogged accounting systems, and overpayments by automatic teller machines. Computer errors could also cause injuries to persons and death through failures in medical equipment, air navigation systems, industrial equipment, or power plants, and one can even imagine that a malfunction in a defence surveillance or weapons system could start a war."

In order to make a reasonable assessment of the risk associated with software defects an estimate of the likely number of defects which would occur needs to be made.

2.7.2 How Many Defects Occur?

For a typical software development project, written in "C", with a size of 1000 function points, typically:

"about 1000 errors would be found in requirements; 1,250 in design; 1,750 in the source code; and 600 in the user manuals. Another 400 bugs would be secondary errors or "bad fixes" introduced when a bug repair itself contained a new error. Roughly 5,000 bugs, then, would have to be found and eliminated in this average project."

The following table summarises software defects including those associated with particular life-cycle products, and includes numbers of defects identified prior to delivery; the efficiency rate of removal of defects; and the number of delivered defects in typical software systems:

<table>
<thead>
<tr>
<th>Defect Origin</th>
<th>Potential Defects (per function point)</th>
<th>Removal Efficiency</th>
<th>Delivered Defects (per function point)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>1.00</td>
<td>77%</td>
<td>0.23</td>
</tr>
<tr>
<td>Design</td>
<td>1.25</td>
<td>85%</td>
<td>0.19</td>
</tr>
<tr>
<td>Coding</td>
<td>1.75</td>
<td>95%</td>
<td>0.09</td>
</tr>
<tr>
<td>Documentation</td>
<td>0.6</td>
<td>80%</td>
<td>0.12</td>
</tr>
<tr>
<td>Bad Fixes</td>
<td>0.4</td>
<td>70%</td>
<td>0.12</td>
</tr>
<tr>
<td>TOTAL</td>
<td>5.00</td>
<td>82%</td>
<td>0.75</td>
</tr>
</tbody>
</table>

*Figure 11: Software Defects*

75 "C" is a software programming language,
Based on the above data, for a typical project of (say) 1,000 function points in size, 5,000 defects will be present in the system overall, although with an 82% removal efficiency 750 will still remain at delivery. Although many of these are likely to be minor or associated with non-code products, obviously any drafter of information technology contracts needs to ensure that firstly, adequate protection is provided in the event of damage caused by such a defect; and adequate flexibility is available in that contract to ensure that fixes can be made.

2.8 SYSTEMS INTEGRATION CONTRACTS

For a large system in particular, one of the primary phases of a software development lifecycle is the Integration Phase. Due to the size, complexity and cost of many large information technology projects, the "systems integration" function, whether it is provided in-house, or via specialist systems engineering organisations, appears to constitute a critical focal point for the success or otherwise of the projects. The overall role of the systems integrator is generally to provide the expertise, framework and infrastructure necessary to ensure that the multiple components usually associated with large information technology systems are developed and integrated in a timely and cost effective manner to achieve customer requirements.

Specifically, the term "System Integration" has been defined as being:

The service which draws together the various elements required to provide a comprehensive information processing solution, through a combination of professional service and expertise.78

From a technical perspective the role of the systems integrator is tending to become more important as software and information technology developments become bigger and more complex and involve large numbers of organisations and individuals.

Naturally the issues identified in previous Sections need to be taken into account by systems integrators however, in addition, a number of additional problems also occur:79

- estimating how long systems integration will take can be very difficult;
- major problems often appear at the systems integration phase because this is essentially the first time that all components of the system are connected together;
- extensive de-bugging is needed, which in itself can be difficult and time consuming;
- all the above problems appear at the end of the project, by which time schedule overruns are likely to introduce additional pressure.

Due to the significant problems which potentially face systems integrators therefore, from a legal perspective it is essential that systems integration contracts are flexible enough to accommodate changing requirements while at the same time ensuring an appropriate level of contractual certainty.80

---

78 Restricted Panel of Systems Integrators Request for Tender issued 2 December 1991 by the Commonwealth Department of Administrative Services, p4; as cited in: Henty, K. Systems Integration Contracts, Proc NSW SCL 9 (1992) 1
79UTS Campus Thomson Software Verification Validation and Test, 1996 (Course Notes)
80 Examples of systems integration contracts where problems have occurred include: Madeley Pty. Limited (via The Venture Group) v Touche Ross & Co. (Unrep), Sup. Ct. of Vic, McGarvie J, 21 Dec 1989, No. 498 of 1988; and Westsub Discounts Pty Limited v Idaps Australia Limited (1990) 17 IPR 185.

Paul P. Parnell 27
2.9 CONCLUSION

Information technology development and systems integration work is already a major industry both within Australia and overseas:

"The $14 billion IT outsourcing and consulting world wide are growing at 15-20% per year, and make up the fastest-growing sector of the Australian IT marketplace. Last year the Australian outsourcing market was valued at $487 million, a 67% increase on the previous year, according to the analyst International Data Corporation (IDC). The company estimates that by the year 2000 this will be close to $1.4 billion. In Australia the total IT external services market - which includes software design and development, systems integration, consulting, systems management and training, and education - is estimated to be a $2.4 billion business."  81

A major software development or integration task can be technically very complex and involve the work of many organisations and developers, some of which are isolated geographically.

Systems Integration contracts obviously play a major part in the development of such projects. The above discussion has briefly outlined some of the issues which are relevant to major software developments and accordingly, those which systems integration contracts need to take into account. The discussion has been by no means exhaustive however it is reasonably clear that major software undertakings, despite the formality of a systems engineering discipline, operate in an environment of continual change and uncertainty. Legal and contractual mechanisms which can cope with this are essential to the success of these endeavours.

3. Contract Interpretation and Related Issues

The objective of this chapter is:

- to discuss the legal principals applicable to information technology contracting (including a number of related legal concepts such as estoppel) which affect legal rights and obligations
- to establish a foundation for following chapters where the practical application of these principals will be discussed

3.1 INTRODUCTION

This chapter is intended to provide a statement of the current law in relation to information technology systems integration contracts, primarily focusing on contract interpretation issues as well as the effect of the conduct of the parties on their legal rights and obligations.

3.2 THE PRE-CONTRACTUAL ENVIRONMENT

3.2.1 Introduction

Both before and after any contract is signed there is usually a significant degree of interaction between the parties involved, and information technology contracts are no exception. The pre-contractual interaction can take many forms ranging from advertising literature associated with a product or service; to invitations to tender for a system and subsequent responses; to the final negotiations between the contracting parties prior to contract signing.

As an example of the extent of such pre-contractual interaction, some large information technology projects have seen the buyer fund one or more companies to provide tender responses for the provision of an information technology based system. This indicates that under some circumstances, the amount of initial interaction between the parties is acknowledged to be so great that the buyer is prepared to pay multiple tenders for such pre-contractual involvement.

Systems integration contracts can involve many parties and as well as a complex mix of technologies which will eventually need to be integrated into a functioning unit.

---

82 This was the case when the Australian Commonwealth Government funded two companies on a tender short-list to provide detailed specifications for the Collins Class submarine acquisition: “Introduction of Underwater Platform Technology (An Australian Perspective)” A paper by Commodore Geoff Rose AM RAN, Collins Class Project Director, for the Defence Asia Forum 1997 at the World Trade Centre Singapore 16 January 1997 http://www.navy.gov.au/oldsite/1news/ccpd16-1.html

Paul P. Parnell
Accordingly, the pre-contractual interaction between the parties can be complex, particularly if a large number of "components" from different suppliers need to be integrated.

In relation to pre-contractual dealings between the parties, the law has developed a number of methods which can be used to ensure that the contracts which are brought into existence are done so by fair means. In particular, the following topics are relevant:

- misrepresentation;
- duress;
- undue influence;
- unconscionable contracts;
- estoppel;
- collateral contracts; and
- representations which become terms of a contract.

From a systems integration point of view, all of the above issues may become relevant in a commercial context. Honest and accurate representations made before the contract comes into existence may induce a party to enter into that contract. In such cases, the question arises as to whether such representations actually become part of the contract, and if so, under what circumstances. In addition, even if the representations do not eventually become part of the contract, other considerations such as estoppel need to be considered (as the party making the representation may be prevented from denying the assertions). In addition, the question of collateral contracts needs to be considered because even if the representations do not become part of the main contract, they may (under certain circumstances) form a collateral contract.

3.2.2 Misrepresentation

3.2.2.1 What constitutes misrepresentation?

In general, four elements are necessary to show misrepresentation - a false statement, of fact, directed to the person mislead, which was intended to induce the contract:

*False statement.*

A representor needs to make some positive representation in order for it to be classed as false. As an example, at common law, mere silence is not enough to constitute a false statement for the purposes of misrepresentation. There are however some exceptions to this rule in relation to silence. For example, if a true statement subsequently becomes untrue, silence may constitute a misrepresentation; if the representor discovers that the statement made was false and fails to correct it; if by silence, there is a distortion of a previously made positive statement, where the parties are in a fiduciary relationship, or if the contract is *Uberrimae Fidei*.

---

[83] With v O'Flanagan [1936] 1 Ch 575; Davies v Provincial Marine Insurance Co. (1878) 8 Ch.D. 469, at 475 per Fry, J.: "... if a statement has been made which is true at the time, but which during the course of the negotiations becomes untrue, then the person who knows that it has become untrue is under an obligation to disclose to the other the change of circumstances."

[84] Davies v Provincial Marine Insurance Co. (1878) 8 Ch.D. 469, at 475 per Fry, J.: "Thus, for instance, if one of the negotiating parties has made a statement which is false in fact, but which he believes to be true and which is material to the contract, and during the course of the negotiation he discovers the falsity of that statement, he is under an obligation to correct his erroneous statement;"

[85] Dimmock v Hallett (1866) L.R. 2 Ch. 21


[87] Gordon v Gordon (1821) 3 Swans. 400; 36 E.R. 910
From the point of view of larger systems integration contracts, it should be noted that the legal relationship between some or all of the parties may include fiduciary elements. For example, two companies who enter into a partnership or joint venture arrangement to perform systems integration work may have fiduciary obligations toward each other. If that is the case, then in some circumstances, silence may constitute misrepresentation.

...of fact...

The statement should be fact not opinion. An example of the operation of this rule can be seen in Bisset v Wilkinson. There are exceptions however to the rule against statements of opinion. If the person making the representation lied (i.e. never actually held that opinion); if a reasonable person could not have held it; if it could be implied that the representer knew of facts which supported the opinion; or where the representer should have known the facts (or could more readily obtain the facts than the other party), then the statement of opinion may be sufficient to constitute misrepresentation.

From the point of view of information technology contracts, uncertainty can be a fact of life, particularly for the larger systems integration work. The uncertainty can be particularly prevalent in areas such as schedule, quality and cost. As a consequence of cases such as Smith v Land & House Property Corp. (discussed above) any opinion which is given about schedule, quality, cost estimates, or other factors could imply that the representer had facts to support that opinion, when this may not be the case. The consequence being therefore that a pre-contractual opinion may constitute misrepresentation (subject to the other elements of the action being met).

A statement of intention is also generally not enough to give rise to an action for misrepresentation unless the promise has become a term of the contract. For example, a statement made by a software developer that the software is intended to be efficient enough to provide screen updates about every second will not necessarily lead to an action in misrepresentation if that is not the case. In some cases however a statement of future intention has been held to be a misrepresentation of an existing part. This can be seen in Edgington v Fitzmaurice where Cotton LJ at 479-480 found that statements made in a prospectus were statements of intention but were also misrepresentations of fact because company directors had never intended to carry out the statements made; and also in JJ Savage & Sons Pty Ltd v Blakney where the High Court of Australia found that a pre-contractual statement relating to the speed of a boat to be purchased,

---

88 Equity applies special obligations of trust and confidence to one or more of the parties under particular relationships; for example see Evans, Michael: Outline of Equity and Trusts. Butterworths, 1998, Chapter 5.

89 "The relation between partners is, of course, fiduciary. Indeed, it has been said that a stronger case of fiduciary relationship cannot be conceived than that which exists between partners. 'Their mutual confidence is the lifeblood of the concern. It is because they trust one another that the business goes on' (per Bacon V.C. in Helmore v Smith (1868) 35 Ch.D. 436 at 444): Cassels v Stewart (1881) 6 App. Cas. at 79.

90 Bisset v Wilkinson (1927) A.C. 177. In this case it was held that an estimate of the number of sheep capable of being run on a property was held to be merely an honest statement of opinion rather than a representation of the true capacity of the land. On the facts of that case, it was held that the purchaser should have been aware of the true figure.

91 For example, see: Yorkville Nominees Pty Ltd (in Lq) v Lissenden (1986) 160 CLR 475. This was an insurance case where although the person signing the insurance policy did not know of a risk, another partner of the firm did know of a material risk. Misrepresentation was found in this case.

92 For example, see the Insurance Contracts Act Ch. 1884, Section 26(1)

93 Smith v Land & House Property Corp (1884) 28 Ch.D. 7, at 15 per Bowen, J. "But if the facts are not equally known to both sides, then a statement of opinion by one who knows the facts best involves very often a statement of a material fact, for he impliedly states that he knows facts which justify his opinion."

94 Brown v Raphael (1958) Ch. 636


96 Edgington v Fitzmaurice (1885) 29 Ch.D. 459

97 JJ Savage & Sons Pty Ltd v Blakney (1970) 119 CLR 435
was a misrepresentation despite the intention of the maker of the statement to guarantee its truth.

A statement of law also cannot usually be used to found an action in misrepresentation. It will be allowed however under certain circumstances such as if the statements contain both law and facts; if the representor deliberately states the law incorrectly; if the statements refer to private rights as opposed to public rights (e.g. Common law, statutory rights); or where the person making the statement has a superior knowledge of the law and, he knows that the other party will rely on that fact.

"[T]he basis of the approach to statements of law is that one can only ever express an opinion as to the law on an issue until a court adjudicates on it."  

Some commentators have criticised this rule that a statement of law cannot be used to found an action in misrepresentation, saying that the "true position is that a statement will not be a basis for relief if it was not, or ought not to have been, relied on, and that its classification as one of law or of fact is irrelevant." In line with this, the High Court of Australia in David Securities Pty Ltd v Commonwealth Bank of Australia found no distinction between statements of fact and law in specific circumstances (recovery of money paid in mistake).

...directed to the person misled...

Only parties to whom a statement was communicated either directly or indirectly as the intended recipient can bring an action for misrepresentation. As an example of this, in Peek v Gurney, the directors of a company issued a prospectus which contained misrepresentations intended to induce the public to invest. It was held by Lord Chelmsford (at 399-400), that the plaintiff who purchased shares on the open market, not as a result of the prospectus, could not succeed in an action for misrepresentation. The reason was that "... there must be something to connect the directors making the misrepresentation with the party complaining... [in order that]... the parties in one way or another are brought into direct communication."

In a large systems integration contract where many parties may be involved, the decision in Peek v Gurney has implications because generally only the party to whom the misrepresentation was communicated will be able to bring an action for misrepresentation. The other parties involved in the project may have no claim. If the statement was made negligently however then liability may be found through an action in tort, where of course there is no restriction in relation to whom the misrepresentation was directed. We will see in the following two chapters (which discuss the new legal concept of micro-agreements) that in large information technology developments, a considerable amount of interaction takes place between the parties both between individuals as well as between organisations.

The important point here is that, provided there is no negligence involved, misrepresentations can potentially be made which have a significant affect on the outcome of a project while at the same time giving no remedy (through misrepresentation) to the parties affected. The next chapter provides a set of case studies which illustrate

---

98 Beattie v Lord Ebury (1872) L.R. 7 Ch. App. 777 at 800-802 per Mellish L.J.
101 David Securities Pty Ltd v Commonwealth Bank of Australia (1992) 175 CLR 353
102 Peek v Gurney (1873) L.R. 6 H.L. 377
103 Hedley Byrne & Co. Ltd. v Heller & Partners Ltd. (1964) AC 465
how seemingly minor interaction between two parties to a large development can significantly affect the outcome.

Despite the above however, an example where indirect communication was held to be sufficient, in Commercial Banking Company (Sydney) v R.H.Brown & Co\(^{104}\), it was held that even though the communication was not direct, it was made with the knowledge and intention that it would be passed on to the person who eventually acted upon it.

..intended to induce the contract.

It must be shown that the statement induced the contract. Some specific instances where a statement has been held not to have induced the contract include firstly, when the statement is not related or material to the contract (e.g. Nicholas v Thompson\(^{105}\)); where the person making the statement was not aware of the representation (e.g. Horfall v Thomas\(^{106}\)); where the person to whom the statement was made actually knew that representation was false (e.g. Redgrave v Hurd \(^{107}\)); and where the representee does not take action based on the representation (e.g. Attwood v Small\(^{108}\)). The intention to induce the contract is also essential\(^{109}\) and the representee must have reasonably relied on the statements.

In the context of large information technology developments we saw in the previous chapter that accurate estimates of cost and schedule can be difficult if not impossible to produce early in the development cycle. Despite this, any information technology contract invariably has cost and schedule related obligations. In order to arrive at a "meeting of the minds" for such contracts naturally pre-contractual negotiations need to take place and it is at that point when the estimates of cost and schedule are used to develop the contractual framework. Any statement which is made with the intention of inducing the other party to enter into a contract is therefore potentially open to a claim of misrepresentation (provided all the elements of the action are present). It is submitted that cost and schedule estimates associated with large information technology contracts can be a minefield in this regard.

3.2.2.2 Types of Misrepresentation.

Depending upon the intention and knowledge at the time the misrepresentation was made, it can generally be classified into one of three different types of misrepresentation - fraudulent, negligent, or innocent. The difference between these is essentially the degree of culpability and the remedies available. A discussion on remedies is provided in the following chapter - "Micro-Agreements" Part 1: Concepts and Case Studies. The following paragraphs however provide a brief outline of these three types of misrepresentation.

3.2.2.2.1 Fraudulent Misrepresentation.

In Derry v Peek\(^ {110}\), Lord Hershelle said that "fraud is proved when it is shown that a false representation has been made (1) knowingly, or (2) without belief in its truth, or (3) recklessly, careless whether it be true or false".

\(^{104}\) Commercial Banking Company (Sydney) v R.H.Brown & Co. (19;2) 126 C.L.R. 337

\(^{105}\) Nicholas v Thompson [1924] V.L.R. 554

\(^{106}\) Horfall v Thomas (1862) 1 H.&C 90; 158 E.R. 813

\(^{107}\) Redgrave v Hurd (1881) 20 Ch.D. 1

\(^{108}\) Attwood v Small (1838) 6 Cl & Fin. 232; 7 E.R. 684

\(^{109}\) Smith v Chadwick (1884) 9 App. Cas. 187; see also Peek v Gurney (1873) LR 6 H.L. 377 at 411 for an illustration of this principle

\(^{110}\) Derry v Peek (1889) 14 App. Cas. 337
In relation to contract law, the primary remedy available is rescission. This is brought about by an election of the innocent party, and notice of that election to the fraudulent party\(^{111}\). Alternatively, the innocent party may also make an application to the court for an order of rescission\(^{112}\).

Damages may be available through the tort of "deceit", although damages will not generally be available through contract because the false statement has not usually become a term of the contract, and thus cannot be breached. If the statement has become a term then damages may be available. In that instance, the innocent party can elect to either rescind the contract for misrepresentation or sue for damages for breach, but not both\(^{113}\).

Regarding the question as to whether damages should be claimed through tort or contract, the tort measure aims at putting the innocent party in the position he had been prior to the tort being committed, while the contract measure examines the position the innocent party would have been in if the contract had been performed.

As an example of how fraudulent misrepresentation may be relevant in the context of large information technology projects, if one supplier gives an estimate for the delivery schedule in pre-contractual negotiations, then if it can be proved that it derived those schedules carelessly or recklessly, an action for fraudulent misrepresentation may possibly be available to the other party if those schedules are not met.

But wait! – aren’t software projects routinely delivered late\(^{114}\)? Estimating software development schedules can be notoriously difficult, particularly for large complex projects. In order to claim fraudulent misrepresentation, an aggrieved party (such as a buyer of software that was delivered late), may try to claim that the supplier of that software carelessly or recklessly developed (incorrect) schedule estimates. To succeed in such an action the plaintiff (i.e. the buyer) would need evidence of such carelessness or recklessness. As a practical point, to avoid such an action, a supplier should make an effort to generate realistic schedule estimates from quantifiable data or models available in the industry\(^{115}\).

3.2.2.2 Negligent Misrepresentation.

The elements of negligent misrepresentation are that (1) if information or advice is given by a person, and (2) that person represents himself as being capable of giving that advice, or (3) realises (or ought to realise) that the other party will rely on the information, and (4) the other party does reasonably rely on that advice to his detriment, then (5) if the advice or information was incorrect the representor will be liable (MLC Assurance Co. v Evatt\(^{116}\), affirmed in L. Shaddock & Associates v Council of the City of Paramatta\(^{117}\) and

---

\(^{111}\) If notice is impossible however, this requirement may be waived (Car & Universal Finance Co. Ltd. v Caldwell [1965] 1 Q.B. 525

\(^{112}\) This can be necessary, for example, where the other party is attempting to obtain an order of specific performance, or has refused to return property etc.

\(^{113}\) In Atali v Kruger (1955) 94 CLR 216 at 222, Dixon C.J., Webb, Kitt, and Taylor J.J. said: "on the footing that the contract had been induced by fraudulent misrepresentation the respondent had a choice. He might sue for damages for breach of warranty, for the statement formed one of the terms of the contract and was not only a representation; but he could not do this and rescind for misrepresentation".

\(^{114}\) "When time is a critical performance metric, most software organizations fare poorly because software projects are routinely late and over budget": Blackburn, Joseph D., Scudder, Gary D., & Van Wassenhove, Luk N.: "Software Development: A Global Survey of Software Developers", IEEE Transactions on Software Engineering, Vol. 22, No. 12, December 1996, pp. 875-885

\(^{115}\) One example of a cost estimation model is "COCOMO" (Constructive Cost Model) discussed in Bohem, B.; Software Engineering Economics, Prentice Hall PTR, 1981.

\(^{116}\) MLC Assurance Co. v Evatt (1968) 122 CLR 556

\(^{117}\) L. Shaddock & Associates v Council of the City of Paramatta (1981) 150 CLR 225

---

Paul P. Pamell

34
San Sebastian Pty. Ltd. v Minister Administering the Environmental Planning and Assessment Act 1979 [118].

Remedies for negligent misrepresentation are similar to fraudulent misrepresentation, that is, rescission, or damages through tort, or if the representation has become a term, damages through contract. If an election to rescind is taken, notice to rescind must be given by the innocent party although unlike remedies for fraudulent misrepresentation, generally there are no exceptions to this rule.

3.2.2.2.3 Innocent Misrepresentation.

Statements made which are false, but were made with the honest belief that they were true, are the subject of innocent misrepresentation. As for the other types of misrepresentation, rescission is available upon election and the giving of notice. If the misrepresentation has become a term of the contract then damages for breach may be available as an alternative to rescission.

In examining the question of whether an innocent misrepresentation has become a term of the contract, Denning L.J. found in Dick Bentley Productions Ltd. v Harold Smith (Motors) Ltd [1965] 2 All E.R. 65 that:

"...if a representation is made in the course of dealings for a contract for the very purpose of inducing the other party to act on it, and it actually induces him to act on it by entering into the contract, that is prima facie ground for inferring that the representation was intended as a warranty. ...But the maker of the representation can rebut this inference if he can show that it really was an innocent misrepresentation, in that he was in fact innocent of fault in making it, and that it would not be reasonable in the circumstances for him to be bound by it."

In other words, Denning L.J. appears to be saying that a misrepresentation which is truly innocent does not become a term of the contract. This decision has been criticised and it is submitted that under some circumstances it may lead to injustice. The end result of Denning L.J. 's approach is that the party who relied on the misrepresentation may be disadvantaged, while the party who made the representation (albeit innocently) is not bound by it. In the Supreme Court of South Australia, Zelling J. quoting another viewpoint said:

"It is not the function of the law of contract to absolve a defendant if he has behaved as a reasonable man would have done, but to hold him to his promises and make him answerable in damages if he has not. The issue whether an assertion is to be construed as a legally binding promise cannot depend upon whether the man who made it was well-meaning or irresponsible, saint or sinner". [119]

From a practical viewpoint it should be recognised from the above cases that, even if made innocently, a misrepresentation may become a term of the contract and thus damages for breach may be available.

3.2.2.3 The Remedy of Rescission.

Rescission is one of the major remedies for misrepresentation. This remedy is founded in equity and as such, is subject to a number of constraints. Firstly, restitution, as an equitable remedy requires substantial (although not precise) restitution, although the court may order financial adjustments to be made. Factors which have been held to prevent

substantial restitution include substantial changes to the subject matter of the contract, and material changes to the position of the parties.

Secondly, the innocent party may elect to avoid or affirm the contract. An election to affirm can be made by actual communication to that effect or by action consistent with the affirmation. Mere inaction may imply affirmation but does not necessarily do so. Likewise, a mere lapse of time, by itself, does not prevent the innocent party from rescinding, although if the innocent party who knows he has a right to rescind does not, or if the period of time is large (e.g. *Leaf v International Galleries*), then the right may be lost.

Thirdly, if an innocent third party has acquired rights in good faith for value without notice, then the right to rescind is lost.

Fourthly, the "rule in Seddon's case" says that "where the conveyance has been executed... a court of equity will set [it] aside only on the ground of actual fraud". In *Leason Pty. Ltd. v Princes Farm Pty. Ltd.* [1983] 2 NSWLR 381 at 387, Helsham J. in the NSW Supreme Court applied the rule to allow rescission for innocent misrepresentation relating to the sale of goods.

### 3.2.2.4 Legislation Relating to Misrepresentation.

#### 3.2.2.4.1 Trade Practices Act 1974 (Cth).

Section 52 is the primary provision in relation to misrepresentation, although Sections 51A, 53, 53A, 53B, 55, 55A, and 59 are also relevant in relation to more specific types of contracts. Section 52(1) says that "a corporation shall not, in trade or commerce, engage in conduct that is misleading or deceptive or likely to mislead or deceive".

The main difference between this provision and the common law position is that Section 52 is very wide. For example, silence, misstatements of law, and misstatements of opinion or intention may be construed as misleading or deceptive under the Act. At common law, the misrepresentation must have actually misled whereas with Section 52, it is enough if the conduct was "likely to mislead".

Remedies under the *Trade Practices Act, 1974* are principally injunction, damages, and ancillary remedies such as an order varying the contract, refusing enforcement, directing refunds or the return of certain property, directing repair or replacement, or payment for any loss or damage. As such, the *Trade Practices Act, 1974* provides for a greater variety of remedies than that available at common law.

#### 3.2.2.4.2 Fair Trading Act

In Australia, state legislation exists which is equivalent to the Commonwealth *Trade Practices Act, 1974*, Section 52. The state *Fair Trading* Acts have similar provisions to the *Trade Practices Act, 1974* but avoid the constitutional limitations of that Act which applies only to corporations.

---

120 *Leaf v International Galleries* [1950] 2 K.B. 86
121 Seddon v North Eastern Salt Co. [1905] 1 Ch. 326, applying Lord Campbell's statement in *Wilde v Gibson* (1848) 1 HLC 605 at 633; 9 E.R. 897 at 909.
122 *Trade Practices Act, 1974* Section 80
123 *Trade Practices Act, 1974* Section 82
124 *Trade Practices Act, 1974* Section 87
125 *Fair Trading Act 1987* (NSW), Section 42; *Fair Trading Act 1985* (Vic), Section 11; *Fair Trading Act 1989* (Qld), Section 38; *Fair Trading Act 1987* (SA), Section 56; *Fair Trading Act 1987* (WA), Section 10; *Fair Trading Act 1990* (Tas), Section 14; *Fair Trading Act 1992* (ACT), Section 12; *Consumer Affairs and Fair Trading Act 1990* (NT) Section 42.
Under the *Fair Trading Acts*, remedies available include an action for damages\(^\text{126}\), orders voiding\(^\text{127}\), varying\(^\text{128}\), or refusing to enforce the contract\(^\text{129}\).

In order to establish a claim under the Act, the plaintiff needs to show firstly, that there was a misrepresentation: *Taco Company of Australia v Taco Bell Pty. Ltd.*\(^\text{130}\); and secondly, that there is a causal connection between the misleading conduct, and the loss suffered (*Brown v Jam Factory Pty. Ltd.*\(^\text{131}\)). It is not necessary that there be an intention to deceive or mislead (*Hornsby Building Information Centre v Sydney Building Information Centre*\(^\text{132}\)) and in addition, the conduct of the defendant as a whole can be taken into account\(^\text{133}\).

3.2.3 Duress

At law, a claim of duress can be made if a degree of pressure is applied such that a reasonable person is coerced to contract on particular terms. The primary question relates to the nature and extent to which the innocent party is pressurised.

Only illegitimate pressure will provide an action for duress. Legitimate pressure, such as mere commercial pressure will not (e.g. *Smith v William Charlick Ltd.*\(^\text{134}\)). The party complaining therefore needs to show that illegitimate pressure was applied, and also, that the pressure was of a nature which could not be resisted by the weaker party. There are essentially three categories of duress - to the person, to goods, and economic duress.

In relation to information technology contracts, although it is possible that duress to the person may occur, it is more likely that duress to goods, or economic duress will be relevant. The following paragraphs provide a discussion of the different types of duress.

### 3.2.3.1 Duress to the Person.

Duress to the person can occur through actual violence, threatened violence or actual or threatened imprisonment. Actual violence to the person which causes a contract to be induced will enable the innocent party to set aside the contract as it was not brought into existence by the exercise of free will. Threatened violence can also bring a claim of duress. The party complaining must show that the threat was made, and also that the threatened violence could cause bodily harm or death (*Barton v Armstrong*\(^\text{135}\)). The court will also look at the surrounding circumstances apart from the words, including the relevant positions of the parties.

### 3.2.3.2 Duress to Goods.

When one party unlawfully seizes, damages, destroys or detains the goods of another party, or makes threats to do so, then a claim of duress to goods may be made. Traditionally duress to goods was not enough to allow the avoidance or a contract as the party coerced still retained his free will (*Skate v Beale*\(^\text{136}\)). One of the key issues then

---

126 For example, see: *Fair Trading Act 1987 (NSW)*, Section 68
127 For example, see: *Fair Trading Act 1987 (NSW)*, Section 72(5)(a)
128 For example, see: *Fair Trading Act 1987 (NSW)*, Section 72(5)(b)
129 For example, see: *Fair Trading Act 1987 (NSW)*, Section 72(5)(c)
130 *Taco Company of Australia v Taco Bell Pty. Ltd.* (1982) 57 ALJR 109;
131 *Brown v Jam Factory Pty. Ltd.* (1981) 35 ALR 79 at 88
132 *Hornsby Building Information Centre v Sydney Building Information Centre* (1978) 140 CLR 216 per Stephen J at 223
134 *Smith v William Charlick Ltd* (1924) 34 C.L.R. 38 per Starke J. at 70
135 *Barton v Armstrong* [1973] 2 NSWLR 598 at 606, Jacobs J. A. said that "The duress or intimidation must consist in threats of violence calculated to cause fear or loss of life or of bodily harm... or [of] unlawful imprisonment... to one party or his or her husband or wife or child".
136 *Skate v Beale* (1841) 11 Ad.&E. 983 at 990; 113 E.R. 688 at 690, where it was said that "an agreement is
appears to be whether the person subject to the coercion had a real alternative but to submit\textsuperscript{137}.

Some cases have found that in all the circumstances the person subjected to the coercion had no real alternative (Astley v Reynolds\textsuperscript{136}). In "The Siboen"\textsuperscript{139}, Kerr J. (obiter at 336) gave the example that:

"if I should be compelled to sign a lease or some other contract for a nominal but legally sufficient consideration under an imminent threat of having my house burnt down or a valuable picture slashed, though without any threat of physical violence to anyone, I do not think that the law would uphold the agreement. I think that a plea of coercion or compulsion would be available."

This view has been applied in other cases (e.g. Universe Tankships Inc. v International Transport Workers Federation\textsuperscript{140}) and detinue and conversion may also be relevant in some cases.

From the point of view of systems integration contracts, it may be possible to make a claim of duress to goods if, for example, the system integrator refused to release the completed system unless the customer agreed to enter into a maintenance agreement\textsuperscript{141}.

3.2.3.3 Economic Duress.
How far should the law go in protecting parties from normal business pressures rather than forcing them to "stand up to threats when they are made"\textsuperscript{142}?

Where one party makes threats against another party in relation to economic interests, and that duress influences the formation of a contract, then a claim of economic duress may be available. In Universe Tankships Inc. v International Transport Workers Federation\textsuperscript{143}, two elements of economic duress were identified - firstly, pressure which amounts to a compulsion of the will of the party coerced; and secondly, the illegitimacy of that pressure:

"There must be pressure, the practical effect of which is compulsion or the absence of choice. Compulsion is variously described in the authorities as coercion or the vitiation of consent. The classic case of duress is, however, not the lack of will to submit but the victims intentional submission arising from the realisation that there is no other practical choice open to him. This is the thread of principal which links the early law of duress (threat to life or limb) with later developments when the law came also to recognise as duress first the threat to property and now the threat to a man's business or trade."

\textsuperscript{138} Astley v Reynolds (1731) 93 E.R. 839
\textsuperscript{139} "The Siboen" and "The Sibotre"[1976] 1 Lloyd's Rep. 293
\textsuperscript{140} Universe Tankships Inc. v International Transport Workers Federation[1983] 1 A.C. 336 at 400
\textsuperscript{141} In practice however if this were to occur, the Exclusive dealing provisions of the Trade Practices Act, 1974 [e.g. Section 47(2)(d)] would be likely to provide relief also.
\textsuperscript{142} Altyah, P.S. "Duress and the Overborne Will Again", (1983) 99 LQR 353 at 356
\textsuperscript{143} Universe Tankships Inc. v International Transport Workers Federation[1983] 1 A.C. 336
\textsuperscript{144} these two elements have been expanded into an approach having three elements in Carter, J.W. & Harland, D.J.: Contract Law in Australia, Third edition, Butterworths, 1996, at p.468 however it is considered that the judicial approach having two elements is simpler to understand and suitable for the present purposes.
\textsuperscript{145} Universe Tankships Inc. v International Transport Workers Federation[1983] 1 A.C. 336 at <f10> per Lord Scarman
Regarding the first element, in *North Ocean Shipping*\(^{146}\), it was regarded that the first test was satisfied if there existed no reasonable alternative for the party subject to the duress. In *Crescendo Management Pty. Ltd. v Westpac Banking Corp.*\(^ {147}\), however, McHugh J.A. (at 46) appeared to focus on the illegitimacy of the pressure rather than the overbearing of the will, thus implying that the reasonable alternative test may not be so exact.

In respect of the second element, although the pressure in *North Ocean Shipping* was unlawful, it was not suggested that unlawful pressure is necessary, although it appears that, as for duress to goods, the pressure must be of a serious nature: *Atlas Express Ltd. v Kafco (Importers & Distributors) Ltd*\(^{148}\).

Mere commercial pressure does not however constitute duress\(^{149}\). The main question from a practical point of view is what constitutes illegitimate pressure in a commercial setting and just how far can the parties go? The answer to this question has an "element of uncertainty"\(^{150}\) although in some respects, cases can be divided into those where a contract exists at the time the pressure is applied, and those where no contract existed at that time. In the respect of the first situation, in *White Rose Flour Milling Co Pty Ltd v Australian Wheat Board*\(^ {151}\) it was found that a threat made by one party to a contract to the effect that it would not perform its obligations under that contract amounted to illegitimate pressure, despite the fact that presumably an action for breach of contract would have been available to the other party. On the other hand, in *Smith v William Charlick Ltd*\(^ {152}\) no illegitimate pressure was found. In that case a wheat board threatened to stop supplying wheat to a flour miller in future unless he paid money for which there was no legal liability to pay:

"it is plain that a mere abstention from selling goods to a man except on condition of his making a stated payment cannot, in the absence of some special relation, answer the description of 'compulsion', however serious his situation arising from the circumstances may be"\(^{153}\)

Despite this, in the *Smith v William Charlick* case, Issacs J (obiter) indicated that under some circumstances the pressure would be illegitimate and lead to a claim of duress. The key issue is whether there is a compulsion of the will to such an extent that payments are not 'voluntary' but 'forced' within the contemplation of the law\(^ {154}\).

Once economic duress is found to exist, that is illegitimate pressure was supplied, the innocent party may claim that the contract is void, although as seen in *North Ocean Shipping Co Ltd v Hyundai Construction Co Ltd (The Atlantic Baron)*\(^ {155}\), this right can be lost through affirmation or ratification\(^ {156}\). In addition, any money paid as a result of the duress may be recoverable under an action in restitution for unjust enrichment\(^ {157}\).

\(^{146}\) *North Ocean Shipping Co Ltd v Hyundai Construction Co Ltd (The Atlantic Baron)* [1979] Q.B. 705

\(^{147}\) *Crescendo Management Pty. Ltd. v Westpac Banking Corp.* (1988) 19 N.S.W.L.R. 40

\(^{148}\) *Atlas Express Ltd. v Kafco (Importers & Distributors) Ltd.* [1989] 3 W.L.R. 389

\(^{149}\) Barton v Armstrong [1976] AC 104 at 121


\(^{151}\) *White Rose Flour Milling Co Pty Ltd v Australian Wheat Board* (1944) 18 ALJ 324

\(^{152}\) *Smith v William Charlick Ltd* (1924) 34 CLR 38

\(^{153}\) *Smith v William Charlick Ltd* (1924) 34 CLR 38 at 66 per Isaacs J

\(^{154}\) *Smith v William Charlick Ltd* (1924) 34 CLR 38 at 66 per Isaacs J

\(^{155}\) *North Ocean Shipping Co Ltd v Hyundai Construction Co Ltd (The Atlantic Baron)* [1979] Q.B. 705 at 720-721

\(^{156}\) In the *Atlantic Baron* case, North Ocean Shipping was held to have affirmed the contract because of the delay of approximately 8 months between when it paid additional money under duress to Hyundai and when it claimed repayment.

In the context of large information technology contracts, a prime contractor or systems integrator may enter into many sub-contracts for components of a system to be developed. Due to the technical nature of the work there may be many difficulties that arise in integrating those components into the overall system. In such a situation it can fairly easily be imagined that a prime contractor may put pressure on sub-contractors to ensure that deliveries are made on time, within budget, and to the desired quality. Chapter 2 discussed how changing requirements and software defects are almost certain to occur in large information technology projects (and following chapters also discuss this in detail in the context of micro-agreements). In such an environment it is conceivable that a systems integrator may, for example, do such things as withhold payment, threaten to limit future work, threaten to terminate a sub-contract unreasonably or impose other pressures.

Whether such pressure amounts to economic duress will depend on the legal issues discussed above, and specifically, whether the pressure amounts to a compulsion of the will of the party coerced, and is illegitimate pressure.

Considering the cases discussed above it can be seen that some factual situations could occur in large information technology contracts that result in a valid claim of economic duress. Based on the White Rose Flour Milling Co Pty Ltd v Australian Wheat Board\(^\text{156}\) case for example, if the systems integrator made a threat to not perform its obligations under a subcontract then that may amount to illegitimate pressure. If the subcontractor performs very specialised work then the market for such work may be limited. Accordingly, there may be no real alternative except for that subcontractor to succumb to the pressure. This may amount to a compulsion of will (e.g. North Ocean Shipping\(^\text{156}\)).

### 3.2.4 Restrictions on a Claim of Duress

In general, a party subject to duress may have a claim against the other party although the following points should be noted.

- the innocent party must have been under the influence of the duress at the time of entering into the agreement
- if the duress was a factor influencing the decision to enter into the contract then that is enough. It is not necessary that the duress be the sole or predominant factor (Bartorv Armstrong\(^\text{160}\))
- duress does not automatically void a contract. It is merely voidable at the election of the coerced party. The right to void the contract may however be lost by affirmation, either express or implied (e.g. North Ocean Shipping Co. Ltd. v Hyundai Construction Co. Ltd.\(^\text{161}\)). It may also be lost by ratification: "if a voidable contract is acted upon, with a knowledge of all the facts... [the]... party who might have avoided it... [cannot]... still avoid it when... it has turned out to his disadvantage" (Ormes v Beadel\(^\text{162}\))
- damages are not available at general contract law as there has been no breach. Damages may however be available through the tort of intimidation.

### 3.2.5 Undue Influence

Undue influence is an equitable concept and arises when a stronger party uses his influence over a weaker party to gain some undue advantage. In Johnson v Butress\(^\text{163}\), Dixon J. indicated that undue influence could be proved by either actual undue influence,

---

\(^{156}\) *White Rose Flour Milling Co Pty Ltd v Australian Wheat Board* (1944) 18 ALJ 324

\(^{156}\) *North Ocean Shipping Co Ltd v Hyundai Construction Co Ltd (The Atlantic Baron)* [1979] Q.B. 705

\(^{160}\) *Bartorv v Armstrong* [1978] A.C. 104 at 119

\(^{161}\) *North Ocean Shipping Co Ltd v Hyundai Construction Co Ltd (The Atlantic Baron)* [1979] Q.B. 705

\(^{162}\) *Ormes v Beadel* (1860) 2 De G.F.&J. 333 at 336; 45 E.R. 649 at 651

\(^{163}\) *Johnson v Butress* (1936) 56 C.L.R. 113
or by a presumption of undue influence. The difference between these is that in the
former, the focus is on the actual transaction, whereas the latter is based on an
antecedent relationship. The antecedent relationship can be of two types: either one of a
class of well recognised relationships, such as parent/child (e.g. Georgis v European Asia
Bank\textsuperscript{164}) or if not well recognised, then it may be established where one party had the
power to dominate the will of the other in such a manner as to lead to transactions of this
kind\textsuperscript{165}.

Although a presumption may exist, in \textit{Westmelton (Vic) Pty. Ltd. v Archer and
Schulman}\textsuperscript{166}, it was found that the presumption could be rebutted if the transaction was
the product of a fully informed independent mind. It also showed that it is not necessary to
show that the weaker party obtained independent advice.

The remedy available for undue influence is rescission, although as with all equitable
remedies, a party who delays in rescinding may be denied relief\textsuperscript{167}.

Focusing on systems integration contracts, it is submitted that it is unlikely that relevant
antecedent relationships will exist to establish a presumption of undue influence. The
reason for this is that in order to find a new category of antecedent relationship which
gives rise to a presumption of undue influence, it is necessary that "one party occupies or
assumes toward another a position naturally involving an ascendancy or influence over
that other, or a dependence or trust on his part"\textsuperscript{168}. In large information systems
developments, corporations usually operate on a commercial basis which includes the
ability to obtain legal advice before committing to contractual obligations. At a corporation
to corporation level it is likely to be difficult to show undue influence. Despite this however
if evidence of actual undue influence is available a claim may be possible. For example, in
very large systems integration contracts it is possible that a single multi-national
corporation has contracted with numerous smaller companies to provide components of a
system to be developed. The resources of the much larger company may be brought to
bear on the smaller companies in the form of undue influence by, for example, pressuring
the smaller company into a specific contract at the risk of losing other work from the larger
company.

\subsection{3.2.6 Unconscionable Contracts}

\subsubsection{3.2.6.1 General Law in Relation to Unconscionable Contracts}

Although in general "equity will not intervene simply to relieve a weaker party of the
adverse effects of a disadvantageous bargain"\textsuperscript{169}, the basis of claiming an unconscionable
dealing is that it may be available "whenever one party, by reason of some condition or
circumstance is placed at a special disadvantage vis-à-vis another and unfair or
unconscionable advantage is taken of the opportunity thereby created"\textsuperscript{170}. We have seen
that undue influence is essentially about inequality of bargaining power. Unconscionable
conduct on the other hand relates to the conduct of the parties\textsuperscript{171}.

\textsuperscript{164} Georgis v European Asia Bank (1989)
\textsuperscript{165} Johnson v Butress (1936) 66 C.L.R. 113 at 119,134; In re Brocklehurst deceased; Hall v Roberts [1978] Ch 14
\textsuperscript{166} Westmelton (Vic) Pty. Ltd. v Archer and Schulman [1982] V.R. 305
\textsuperscript{167} Allard v Skinner (1887) 36 Ch.D. 145
\textsuperscript{168} Johnson v Butress (1936) 66 C.L.R. 113 at 134-135 per Dixon J
\textsuperscript{169} Commercial Bank of Australia v Amadio (1983) 151 CLR 447
\textsuperscript{170} Commercial Bank of Australia v Amadio (1983) 151 CLR 447, per Mason J. at 474
\textsuperscript{171} Bridgewater v Leahy [1988] HCA 68, 22 October 1998 (unrep) per the majority: Gaudron, Gummow, Kirby
J; Gleeson CJ, Callinan J dissenting.

Paul P. Parnell 41
There are two types of unconscionability - substantial and procedural. In equity, the manner in which the bargain is obtained (i.e. procedural unconscionability) is more important than the terms of the bargain itself (i.e. substantial unconscionability), although substantial unconscionability may be used as evidence of procedural unconscionability.

In order to take advantage of the doctrine, the plaintiff needs to show that he was firstly, in a position of special disadvantage, and secondly, the defendant took advantage of this position in order to obtain the bargain: "it is necessary for the plaintiff who seeks relief to establish unconscionable conduct, namely that unconscionable advantage has been taken of the disabling condition or circumstances." Commercial Bank of Australia v Amadio 172.

The classes of "disadvantage" are somewhat difficult to classify however:

"Among them are poverty or need of any kind, sickness, age, sex, infirmity of body or mind, drunkenness, illiteracy or lack of education, lack of assistance or explanation where assistance or explanation is necessary. The common characteristic seems to be that they have the effect of placing one party at a serious disadvantage".173.

Subsequently Mason J. in Amadio174 indicated that the list of categories is not closed.

In a systems integration context it is considered unlikely that a claim of unconscionability could be made as it would be difficult to establish that a position of special disadvantage existed. Despite this, if a prima facie case of unconscionability did exist, then the defendant bears the onus of showing that the transaction was conducted in a fair, just, and reasonable manner.

3.2.6.2 Legislation relating to Unconscionable Transactions.

3.2.6.2.1 Trade Practices Act 1974 (Cth).

In Australia the Trade Practices Act, 1974 provides some statutory requirements related to unconscionability. Specifically, Section 51AA supports the equitable doctrine of unconscionability in the following terms:

Section 51AA(1) "A corporation must not, in trade or commerce, engage in conduct that is unconscionable within the meaning of the unwritten law; from time to time, of the States and Territories".

The term "unwritten law" has been judicially interpreted as meaning, in effect, the equitable doctrine of unconscionability 175. Under this doctrine, as previously discussed, cases such as Blomley v Ryan176, Louth v Diprose177, and Commercial Bank of Australia v Amadio178 indicate that where "a person under a special disadvantage is taken unfair advantage of by the other party to the contract in circumstances where the other party is aware of the special disadvantage"179, then relief may be obtained.

172 Commercial Bank of Australia v Amadio (1983) 151 CLR 447, per Mason J. at 384
173 Blomley v Ryan (1956) 99 CLR 362 at 405 per Fullagar J.
174 Commercial Bank of Australia v Amadio (1983) 151 CLR 447, per Mason J. at 384
176 Blomley v Ryan (1956) 99 CLR 362
177 Louth v Diprose (1992) 175 CLR 621
178 Commercial Bank of Australia v Amadio (1983) 151 CLR 447
179 as discussed by Zumbo, Frank: "Unconscionability Within Commercial Transactions", being seminar materials from a seminar presented in Sydney March 1996, Legal and Accounting Management Seminars Pty.
Despite the potential application of Section 51AA, it is likely to have limited application in commercial contexts due to the requirement to show "special disadvantage", although it is by no means impossible for an "economically powerful organisation" to impose a special disadvantage on weaker entity\(^{180}\).

In addition to Section 51AA, Section 51AB of the Trade Practices Act, 1974 prohibits a corporation acting in trade or commerce, from engaging in conduct which is, in all the circumstances, unconscionable. This Section also lists factors to be taken into account in determining whether such conduct has occurred. Specifically, the Section states:

"(1) A corporation shall not, in trade or commerce, in connection with the supply or possible supply of goods or services to a person, engage in conduct that is, in all the circumstances, unconscionable.

(2) Without in any way limiting the matters to which the Court may have regard for the purpose of determining whether a corporation has contravened sub-section (1) in connection with the supply or possible supply of goods or services to a person (in this subSection referred to as the "consumer"), the Court may have regard to:

(a) the relative strengths of the bargaining positions of the corporation and the consumer;

(b) whether, as a result of conduct engaged in by the corporation, the consumer was required to comply with conditions that were not reasonably necessary for the protection of the legitimate interests of the corporation;

(c) whether the consumer was able to understand any documents relating to the supply or possible supply of the goods or services;

(d) whether any undue influence or pressure was exerted on, or any unfair tactics were used against, the consumer or a person acting on behalf of the consumer by the corporation or a person acting on behalf of the corporation in relation to the supply or possible supply of the goods or services; and

(e) the amount for which, and the circumstances under which, the consumer could have acquired identical or equivalent goods or services from a person other than the corporation....

(5) A reference in this Section to goods or services is a reference to goods or services of a kind ordinarily acquired for personal, domestic or household use or consumption."

The effect of this Section is to prevent a corporation from engaging in unconscionable conduct, in respect of the supply of goods or services for personal, domestic or household use or consumption. In addition, it provides a (non-exclusive) checklist of things which a court may take into consideration when addressing the question of whether unconscionable conduct has taken place.

The problem with Sections 51AA and 51AB of the Trade Practices Act, 1974 is that they are generally designed (or interpreted) to provide protection to consumers in the traditional sense. For example, the reference in Section 51AA to the "unwritten law" effectively brings into play the common law concept of unconscionability, rather than a new legislative meaning of that term. Specifically, the common law concept of...
unconscionability is primarily designed to give relief where a position of special disadvantage exists, such as in situations where there is: "ignorance of material facts known to the other party, illiteracy or lack of education, poverty or need of any kind, age, infirmity of body or mind, drunkenness, lack of assistance or explanation where these are necessary, or emotional dependence."\(^{181}\)

Although it is possible that a position of special disadvantage could exist in a commercial situation, the above common law criteria for establishing unconscionability, as well as Sections 51AA and 51AB of the Trade Practices Act, 1974 do not really encourage use of the concept in a commercial environment.

Attempts have been made however to extend unconscionability into the commercial or business context. As an example, four attempts were made to incorporate a new Section 51AC into the Trade Practices Act, 1974 with the aim of providing relief against either "harsh or oppressive conduct", or "unconscionability" within the business world. The Trade Practices Amendment (Better Business Conduct) Bill, was introduced in 1995, 1996 and again in 1997\(^{182}\) with a focus on "harsh and oppressive conduct". The final form of the legislation was finally enacted in 1998\(^{183}\) and relevant parts are reproduced below:

```
51AC Unconscionable conduct in business transactions

(1) A corporation must not, in trade or commerce, in connection with:
(a) the supply or possible supply of goods or services to a person (other than a listed public company); or
(b) the acquisition or possible acquisition of goods or services from a person (other than a listed public company);
    engage in conduct that is, in all the circumstances, unconscionable.

(2) A person must not, in trade or commerce, in connection with:
(a) the supply or possible supply of goods or services to a corporation (other than a listed public company); or
(b) the acquisition or possible acquisition of goods or services from a corporation (other than a listed public company);
    engage in conduct that is, in all the circumstances, unconscionable.
```

The legislation also provides a non exclusive list of issues with which the court may have regard when considering an application under this Section. Separate lists are included for a supplier\(^{184}\) and acquirer\(^{185}\). These lists mirror each other and for example the court may consider in relation to a supplier the following issues:

\[51AC(3)](a)\] the relative strengths of the bargaining positions of the supplier and the business consumer; and

(b) whether, as a result of conduct engaged in by the supplier, the business consumer was required to comply with conditions that were not reasonably necessary for the protection of the legitimate interests of the supplier; and

---


\(^{182}\) Department of the Parliamentary Library, Bills Digest No. 55 1997-98, Trade Practices Amendment (Fair Trading) Bill 1997

\(^{183}\) Section 51AC was added by Trade Practices Amendment (Fair Trading) Act 1998, Act. No. 36, 1998

\(^{184}\) Trade Practices Act, 1974 Section 51AC(3)

\(^{185}\) Trade Practices Act, 1974 Section 51AC(4)
(c) whether the business consumer was able to understand any documents relating to the supply or possible supply of the goods or services; and
(d) whether any undue influence or pressure was exerted on, or any unfair tactics were used against, the business consumer or a person acting on behalf of the business consumer by the supplier or a person acting on behalf of the supplier in relation to the supply or possible supply of the goods or services; and
(e) the amount for which, and the circumstances under which, the business consumer could have acquired identical or equivalent goods or services from a person other than the supplier; and
(f) the extent to which the supplier's conduct towards the business consumer was consistent with the supplier's conduct in similar transactions between the supplier and other like business consumers; and
(g) the requirements of any applicable industry code; and
(h) the requirements of any other industry code, if the business consumer acted on the reasonable belief that the supplier would comply with that code; and
(i) the extent to which the supplier unreasonably failed to disclose to the business consumer:
   (i) any intended conduct of the supplier that might affect the interests of the business consumer; and
   (ii) any risks to the business consumer arising from the supplier's intended conduct (being risks that the supplier should have foreseen would not be apparent to the business consumer); and
(j) the extent to which the supplier was willing to negotiate the terms and conditions of any contract for supply of the goods or services with the business consumer; and
(k) the extent to which the supplier and the business consumer acted in good faith.

Section 51AC of the Trade Practices Act, 1974 attempts to provide some protection against unconscionable conduct in the context of business transactions. It applies to transactions of up to $1,000,000 and a business consumer includes a corporation but not a publicly listed company. Although Section 51AC has not had much chance to be judicially considered, presumably the intention of parliament was to provide remedies against unconscionable conduct in an area (that is, business transactions) which traditionally has not had much support judicially.

It remains to be seen how the courts will apply this new Section however it seems to be common sense that unconscionable conduct is unconscionable conduct regardless of whether it occurs in a private or business situation. Large information technology projects may involve large numbers of individuals and organisations and accordingly it is possible that in all the dealings which occur in that context, some unconscionable conduct exists — it would be beneficial to the industry as a whole if this conduct could be restrained as much as possible.

3.2.6.2 Fair Trading Acts

In order to overcome the Constitutional limitations of the Trade Practices Act, 1974 (i.e. to apply to other than corporations engaging in trade or commerce), state legislation also exists in (for example) the Fair Trading Act (N.S.W.) s43, and the Industrial Relations Act, 1991 (NSW) Section 275.

---

186 Trade Practices Act, 1974, Section 51AC(9), (10)
187 Trade Practices Act, 1974, Section 51AC(1), (2)
188 also other state acts Fair Trading Act 1985 (Vic), Section 11A; Fair Trading Act 1989 (Qld), Section 39; Fair Trading Act 1987 (SA), Section 57; Fair Trading Act 1987 (WA), Section 11; Fair Trading Act 1990 (Tas), Section 15; Fair Trading Act 1992 (ACT), Section 13; Consumer Affairs and Fair Trading Act 1990 (NT) Section 43. 
189 previously the Industrial Arbitration Act, Section 88F.

Paul P. Parnell
45
3.2.6.2.3 Contracts Review Act 1980 (N.S.W.)

In N.S.W., unjust contracts may be subject to the provisions of the Contracts Review Act, 1980 although remedies are not available to publicly listed companies or the Crown. The term "unjust" is defined to include "unconscionable" in s4(1). Section 9(1) provides that the public interest, as well as all the circumstances of the case may be taken into account by the court. Thus the scope of relief is wider than that at common law. Section 7 and 8 of that Act provides a list of remedies available. The principle remedies under s7 are refusal to enforce any or all provisions of the contract, declaring the contract void, or varying the contract. Section 8 lists ancillary remedies and as such overall, the Act provides a greater range of remedies than available at common law.

3.2.7 Estoppel

_estoppel_, n. Law, a bar or impediment preventing a party from asserting a fact or a claim inconsistent with a position he previously took, either by conduct or by words, esp. where a representation has been relied or acted upon by others.\(^{190}\)

3.2.7.1 The Basic Principle of Estoppel

The basic principle of estoppel is that "the law should not permit an unjust departure by a party from an assumption which he has caused another party to adopt or accept for the purpose of their legal relations. The departure will be unjust if the other person having acted or abstained from acting on the basis of the assumption, would suffer a detriment if the opposite party were afterwards allowed to set up rights against him inconsistent with that assumption."\(^{191}\)

3.2.7.2 Types of Estoppel

Broadly speaking there are two types of estoppel - common law estoppel and equitable estoppel although since the Walton’s\(^{192}\) case there has been effectively a merging of the doctrines to some extent. Despite this it is the intention of this Section to firstly present a discussion of equitable and common law estoppel separately in order to provide an historical perspective of the doctrines; and secondly to show how the law has effectively merged the two doctrines.

In relation to equitable estoppel, in the past a number of different types of equitable estoppel could be clearly distinguished although in recent times the distinction has been somewhat blurred. Specifically, relatively recent cases\(^{193}\) have seen the merger of a number of the equitable doctrines such as proprietary estoppel, promissory estoppel, estoppel by acquiescence and equitable estoppel into one "broad flow of principle."\(^{194}\)

In summarising the law in relation to the above two classes of estoppel, Priestly J. made the following observations. He based these observations on judgements delivered in Walton Stores (Interstate) v Maher.\(^{195}\)

"(1) Common law and equitable estoppel are separate categories, although they have many ideas in common. (2) Common law estoppel operates upon a representation of

---

\(^{190}\) The Macquarie Dictionary, Macquarie Library, 1981


\(^{192}\) Walton Stores (Interstate) Ltd. v Maher & Anor. (1988) 62 ALJR 119

\(^{193}\) e.g. Walton Stores (Interstate) Ltd. v Maher & Anor. (1988) 62 ALJR 110.


existing fact, and when certain conditions are fulfilled, establishes a state of affairs by reference to which the legal relation between the parties is to be decided. This estoppel does not itself create a right against the party estopped. The right flows from the courts decision on the state of affairs established by the estoppel. (3) Equitable estoppel operates upon representations or promises as to future conduct, including promises about legal relations. When certain conditions are fulfilled, this kind of estoppel is itself an equity, a source of legal obligation. (4) Cases described as estoppel by encouragement, estoppel by acquiescence, proprietary estoppel and promissory estoppel are all species of equitable estoppel. (5) For equitable estoppel to operate in circumstances such as those of the present case there must be the creation or encouragement by the defendant in the plaintiff of an assumption that a contract will come into existence or a promise be performed, and reliance on that by the plaintiff, in circumstances where departure from that assumption by the defendant would be unconscionable. (6) Equitable estoppel may lead to the plaintiff acquiring an estate or interest in land; that is, in the common metaphor, it may create a sword. (7) The remedy granted to satisfy the equity (which either is the estoppel or created by it) will be what is necessary to prevent detriment resulting from the unconscionable conduct. 196

Although the statement by Priestly J. above provides a relatively clear and concise statement of the law, further expansion of a number of points is made in the following paragraphs together with a brief discussion of applicability to information technology systems integration contracts where appropriate.

3.2.7.3 Common Law Estoppel

Common law estoppel (sometimes also referred to as estoppel in pais, evidentiary estoppel, estoppel by conduct or estoppel by representation 197) arises when a representation is made as to an existing fact 198.

In relation to systems integration contracts it is possible that representations as to existing fact may be made between the parties. For example, during pre-contractual tender negotiations, an organisation tendering may claim that it owns all intellectual property rights of an existing (similar) system when in fact copyright of some of the components may be in fact owned by independent third parties 199. This situation could lead to common law estoppel being brought into play because the representation as to an existing fact was incorrect. It should be noted that common law estoppel itself is not a source of legal obligation, except "in the sense that the estoppel compels the party bound to adhere to the assumption that the contract exists" 200. The legal obligation arises from the contract which is assumed to come into existence.

Some examples of common law estoppel found in a commercial context include:

"in Laws Holdings Pty Ltd v Short (1972) 46 ALJR 563, a company which had led a supplier of goods to assume that it was the purchaser of goods, in fact received by an associated company of similar name was held bound by a contract between itself and the

196 Silovi Pty Ltd v Barboro (1988) 13 N.S.W.L.R. 466 at 472 per Priestly J.A.
198 This was summed up in: Waltons Stores (Interstate) Ltd v Maher & Anor. (1988) 62 A.L.J.R. 110, 113 per Mason C.J. and Wilson J. as follows: "There is, as Mason and Deane JJ. pointed out in Legione v Hately (1983 152 C.L.R. 406, 432, a long line of authority to support the proposition that, to make out a case of common law estoppel by representation, the representation must be as to an existing fact, a promise or representation as to future conduct being insufficient (Jordan v Money (1854) 5 H.L.C. 185; 10 E.R. 668; Maddison v Alderson (1883) 8 App. Cas. 467, 473)."
199 Such a situation occurred in Accounting Systems 2000 (Developments) Pty Ltd v CCH Australia Ltd (1993) 42 FCR 470, 27 IPR 133, where representations of existing fact were effectively made in the form of certain warranties as to the title of software provided under a contract. In this case however the issue of estoppel was not considered but rather, misleading and deceptive conduct was found under Section 52 of the Trade Practices Act, 1974 Section 52.
supplier which its conduct had led the supplier to assume to exist. And in Spiro v Lintern [1973] 1 WLR 1002; [1973] 3 All ER 319, a husband whose wife had made a contract in her own name for the sale of the husband’s property without his authority was estopped from denying that she had made the contract with his authority.\(^{201}\)

In terms of practical affect, in written contracts, some parts of the document such as “recitals”, “Agreed Statements of Facts”, or “Preamble” may not involve strict rights and obligations under contract law, but may become subject to the doctrine of common law estoppel. For this reason care is needed to be taken when drafting such clauses as well as the main contractual clauses.

### 3.2.7.4 Equitable Estoppel

Perhaps the best way to describe equitable estoppel is to refer to the elements required to establish it. These elements were stated by Brennan J. in Walton Stores (Interstate) Limited\(^{202}\) v Maher as follows:

> “In my opinion, to establish an equitable estoppel, it is necessary for the plaintiff to prove that (1) the plaintiff assumed or expected that a particular legal relationship exists between the plaintiff and the defendant or that a particular legal relationship will exist between them and, in the later case, that the defendant is not free to withdraw from the expected legal relationship; (2) the defendant has induced the plaintiff to accept that assumption or expectation; (3) the plaintiff acts or abstains from acting in reliance on the assumption or expectation; (4) the defendant knew or intended him to do so; (5) the plaintiff’s action or inaction will occasion detriment if the assumption or expectation is not fulfilled; and (6) the defendant has failed to act to avoid that detriment whether by fulfilling the assumption or expectation or otherwise”.

From an historical perspective, promissory estoppel which is one form of equitable estoppel\(^{203}\) evolved over time and had been addressed by the courts in a number of cases. It is generally acknowledged that the doctrine crystallised in Central London Property Trust Ltd. v High Trees House Ltd. [1947] K.B. 130 although earlier cases\(^{204}\) appear to have established the basis of the doctrine. In Australia, Legione v Hateley\(^{205}\) adopted and applied the concept of promissory estoppel, with subsequent cases such as Walton Stores (Interstate) Limited\(^{206}\) v Maher and Commonwealth v Verwayen\(^{207}\) further considering and refining the limits of the doctrine\(^{208}\).

### 3.2.7.5 The Basis of Equitable Estoppel is Unconscionability

In order to understand the doctrine of equitable estoppel, and to subsequently apply it in a modern day commercial context, the basis of the doctrine must be understood. Throughout the modern cases including Walton Stores (Interstate) v Maher\(^{209}\), and Commonwealth v Verwayen\(^{210}\), the focus seems to be on unconscionability. For example:

> “Parties who are negotiating a contract may proceed in the expectation that the terms will be agreed and a contract made but, so long as both parties recognise that either party is at liberty to withdraw from the negotiations at any time before the contract is made, it cannot be unconscionable for one party to do so. Of course the freedom to withdraw may be fettered or extinguished by agreement but, in the absence of agreement, either party


\(^{203}\) Silvoly Pty Ltd v Barban (1988) 13 N.S.W.L.R. 466 at 472 per Priestly J.A.

\(^{204}\) e.g Ramsden v Dyson (1866) L.R. 1 H.L. 128; Hughes v Metropolitan Railway Co. (:577) 2 App. Cas. 439.

\(^{205}\) Legione v Hateley (1983) 152 C.L.R. 406


\(^{207}\) Commonwealth v Verwayen (1990) 170 CLR 394

\(^{208}\) the contribution of these and other cases to the doctrine is discussed in more detail later.


\(^{210}\) Commonwealth v Verwayen (1990) 170 CLR 394
ordinarily retains his freedom to withdraw. It is only if a party induces the other party to believe that he, the former party, is already bound and his freedom to withdraw has gone that it could be unconscionable for him subsequently to assert that he is legally free to withdraw.\(^{211}\)

Note however that breaking a promise in itself is not enough:

"As failure to fulfil a promise does not of itself amount to unconscionable conduct, mere reliance on an executory promise to do something, resulting in the promisee changing his position or suffering detriment, does not bring promissory estoppel into play. Something more would be required."\(^{212}\)

Specifically, the "central principal of the doctrine is that the law will not permit an unconscionable ... departure by one party from the subject matter of an assumption which has been adopted by the other party ... [and] which would operate to that other party's detriment if the assumption be not adhered to."\(^{213}\)

A degree of unconscionability is thus required and this may involve issues of unfairness and injustice or inequitability\(^{214}\). In the Walton\(\text{s}\) case, factors which were taken into consideration when determining whether equitable estoppel came into play included the urgency of the situation, because of the expectation created and acted upon\(^{215}\).

This is particularly relevant in the context of systems integration contracts because quite often, the amount of time and effort necessary to complete the development and integration work can be difficult to estimate. This inaccuracy in estimating time-scales and schedules can often crystallise into an urgent requirement to complete the work to ensure contracted schedules are met. In the Walton\(\text{s}\) Stores (Interstate) v Maher decision, the issue was summed up by Mason C.J. and Wilson J. as follows:

"The mere exercise of its legal right not to exchange contracts could not be said to amount to unconscionable conduct on the part of the appellant. But there were two other factors present in the situation which require to be taken into consideration. The first was the element of urgency that pervaded the negotiation of the terms of the proposed lease... The second factor of importance is that the respondents executed the counterpart deed and it was forwarded to the appellant's solicitor on 11 November. The assumption on which the respondents acted thereafter was that completion of the necessary exchange was a formality..."\(^{216}\).

In the Walton\(\text{s}\) case, the defendant acted in a somewhat reprehensible manner. By its silence, it induced the plaintiff to continue work on the assumption that the defendant was already, or would be bound when this was not the case. A clear case of unconscionable conduct was thus made out in that situation.

If unconscionable conduct is thus acknowledged to be the basis of the doctrine of equitable estoppel, what then constitutes unconscionable conduct? In Walton\(\text{s}\) Stores (Interstate) v Maher decision, when attempting to answer the question as to what conduct may be regarded as unconscionable, Deane J. stated:


\(^{213}\) Commonwealth v Verwayan (1990) 170 CLR 394 at p.444 per Deane J.


Paul P. Parnell
"the question whether conduct is or is not unconscionable in the circumstances of a particular case involves a "real process of consideration and judgement" in which the ordinary processes of legal reasoning by induction and deduction from settled rules and decided cases are applicable but are likely to be inadequate to exclude an element of value judgement in a borderline case."\(^{217}\)

The effect of this statement is that the 'categories' or circumstances giving rise to unconscionable conduct are not closed. Each circumstance should be therefore looked at on its merits while, of course, taking precedents into account as well.

3.2.7.6 What is the Difference between a Contract and an Equity Created by Estoppel?

In considering whether the conduct of a party will form a legal obligation, under some circumstances that obligation will arise under the contract itself, whereas in other circumstances an obligation may emerge through equitable estoppel. In addressing the difference between the above two types of obligation, Brennan J. in the Walton Stores (Interstate) v Maher decision stated that:

"A contractual obligation is created by the agreement of the parties; an equity created by estoppel may be imposed irrespective of any agreement by the party bound. A contractual obligation must be supported by consideration; an equity created by estoppel need not be supported by what is, strictly speaking, consideration. The measure of contractual obligation depends on the terms of the contract and the circumstances to which it applies; the measure of an equity created by estoppel varies according to what is necessary to prevent detriment resulting from unconscionable conduct."

An important practical point to understand from the above distinction is that the remedies which may be available through each of the above causes of action (that is, contract or equitable estoppel) may be different.

3.2.7.7 Equitable Estoppel May Apply Regardless of Whether there is an Existing Contract

The cases have shown that the doctrine of equitable estoppel may be relevant regardless of whether a contract exists\(^{218}\). Some examples of each category are provided in the following paragraphs:

3.2.7.7.1 Where there is an Existing Contract

In D. & C. Builders Ltd. v Rees\(^{220}\), a creditor agreed to accept payment of a lesser sum in satisfaction of the whole amount of the debt, but later tried to claim the remainder as a legal right. In relation to this point, Denning L.J. stated:

"...when a creditor and a debtor enter upon a course of negotiation, which leads the debtor to suppose that, on payment of a lesser sum, the creditor will not enforce payment of the balance, and on faith thereof the debtor pays the lesser sum and the creditor accepts it as satisfaction; then the creditor will not be allowed to enforce payment of the balance when it would be inequitable to do so."


\(^{220}\) D. & C. Builders Ltd. v Rees [1966] 2 QB 617, at 624 per Denning L.J.
The above decision found its basis in *Hughes v Metropolitan Railway*\(^{221}\). In that case, Lord Cairns stated:

"It is the first principle upon which all courts of equity proceed, that if parties, who have entered into definite and distinct terms involving certain legal results, afterwards by their own act or with their own consent enter upon a course of negotiation which has the effect of leading one of the parties to suppose that the strict rights arising under the contract will not be enforced, or will be kept in suspense, or held in abeyance, the person who otherwise might have enforced those rights will not be allowed to enforce them when it would be inequitable having regard to the dealings which have taken place between the parties."

### 3.2.7.7.2 Where there is No Existing Contract

An example of a case where no contract existed, although equitable estoppel was used to avoid the effects of unconscionable conduct was *Walton Stores (Interstate) Limited v Maher*\(^{222}\). In relation to this issue, Mason C.J. and Wilson J. stated that:

"The appellant's inaction, in all the circumstances, constituted clear encouragement or inducement to the respondents to continue to act on the basis of the assumption which they had made. It was unconscionable for it, knowing that the respondents were exposing themselves to detriment by acting on the basis of a false assumption, to adopt a course of inaction which encouraged them in the course they had adopted. To express the point in the language of promissory estoppel the applicant is estopped in all the circumstances from retracting from its implied promise to complete the contract."\(^{223}\)

The issue is important from a systems integration context as detailed negotiations are likely to take place by many parties prior to a formal contract being put in place. Specifically, a systems integrator will need to negotiate with the customer plus subcontractors to ensure all components of a software project are effectively integrated. The Government tendering process is another example of where significant interaction between the parties may occur prior to a contract being formed. In all these situations therefore it is important to understand the consequences of implied promises made during negotiations.

The point to note from the *Walton Stores (Interstate) v Maher* case is that a party can be bound through equitable estoppel regardless of whether a contract is in place.

### 3.2.7.8 The Relationship between Equitable Estoppel and Consideration

In terms of its practical effect, equitable estoppel (and particularly promissory estoppel, which is one form of equitable estoppel), has been seen to encroach on the doctrine of consideration to the extent that there was initially some reluctance to allow the concept of

\(^{221}\) *Hughes v Metropolitan Railway Co.* (1877) 2 App. Cas. 439 at 448 per Lord Cairns. See also *Je Maintiendrai Pty Ltd v Quaglia* (1980) 26 SASR 101

\(^{222}\) *Walton Stores (Interstate) Limited v Maher & Anor.* (1988) 164 C.L.R. 387; 62 A.L.J.R. 110. The facts of this case were that Walton Stores operated a chain of retail stores and negotiated with Maher for the lease of commercial premises upon which a Walton's Store was to be built. Under the arrangement, Maher was to construct and fit-out a special purpose building for Walton's, in the timeframe required by Walton's. Detailed negotiations took place and eventually a "Deed of Agreement for Lease" was sent by Walton's to the Maher together with a covering letter which stated: 'We shall let you know tomorrow if the amendments are not agreed to'. Walton's did not subsequently notify Maher of any objection and on that basis, Maher returned the (now executed) Deed "by way of exchange" to Walton's. At this point in time both parties knew that Maher would need to 'organise labour and order supplies' to meet Walton's schedule. After several months Walton's advised Maher that it did not wish to proceed with the lease. By this time, Maher was however committed to the building work.

equitable estoppel to develop. Mason C.J. and Wilson J. summarised this reluctance in Walton Stores (Interstate) Limited v Maher as follows:

"The principal objection to the enforcement of such a promise is that it would outflank the principles of the law of contract. Holmes J. expressed his objection to the operation of promissory estoppel in this situation when he said: 'It would cut up the doctrine of consideration by the roots. If a promisee could make a gratuitous promise binding by subsequently acting in reliance on it' (Commonwealth v Scituate Savings Bank (1884) 137 Mass. 301),302. Likewise, Sir Owen Dixon considered that estoppel cut across the principles of the law of contract, notably offer and acceptance and consideration: "Concerning Judicial Method" (1956) 29 Australian Law Journal 468, 475. And Denning L.J. in Combe v Combe224, after noting that "The doctrine of consideration is too firmly fixed to be overthrown by a side-wind", said (at 220) that such a promise could only be enforced if it was supported by sufficient consideration".

Despite the above authorities however, the court in Walton Stores (Interstate) Limited v Maher found the solution by applying equitable estoppel only to situations involving unconscionable conduct. Brennan J., in that case, stated:

"But the ... solution to the problem is reached by identifying the unconscionable conduct which gives rise to the equity as the leaving of another to suffer detriment occasioned by the conduct of the party against whom the equity is raised. Then the object of the principle can be seen to be avoidance of that detriment and the satisfaction of the equity calls for the enforcement of a promise only as a means of avoiding the detriment and only to the extent necessary to achieve that object. So regarded, equitable estoppel does not elevate non-contractual promises to the level of contractual promises and the doctrine of consideration is not blown away by a side-wind. Equitable estoppel compliments the tortious remedies available to a party who acts or abstains from acting in reliance on what another induces him to believe."226

3.2.7.9 Equitable Estoppel Can be a Sword as well as a Shield

There is now no bar to the use of equitable estoppel as a course of action in its own right. In other words, a plaintiff does not have to rely on any existing contract, but can commence an action by use of equitable estoppel alone.

"It follows that a plaintiff in the position of the Mahers in the present case is entitled to plead and rely upon the facts giving rise to an operative estoppel by conduct which precludes the defendant from denying the existence of a binding contract for the purpose of affirmatively establishing the foundation for the case to be dealt with on the basis of the assumed fact that there was such a contract. That being so, such an estoppel provides the factual foundation of an ordinary action for enforcement of that "contract" notwithstanding that those facts demonstrate that no binding contract was actually made."227

The significance of this is that the High Court effectively used estoppel as a cause of action in its own right which resulted in an award of damages. Normally the absence of consideration precludes any action being taken through contract and thus precludes any award of damages228.

---

224 Combe v Combe [1951] 2 KB 215

Paul P. Parnell
3.2.7.10 The Practical Effect of Equitable Estoppel

So overall then, what is the practical effect of equitable estoppel? The above cases show that equitable estoppel may operate regardless of whether there is a contract in place or not. That is, the legal relationship of the parties is not relevant to the action in this context. In large systems integration contracts, equitable estoppel may be applicable therefore regardless of whether a contract exists. This is important because usually extensive negotiations take place between the parties involved in systems integration work before any contract is signed. Moreover, it is possible that the parties involved do not intend to enter into a “traditional” systems integration contract per se, but rather may wish to undertake activities under a joint venture, partnership, or other commercial arrangement such as forming a separate corporation. Even though those particular relationships involve contracts which govern the relationship itself, once that relationship is established, there may not be a need for a traditional contract for the development of the system.

An example of such a relationship is where an acquirer and a developer of a system set up a separate corporation which has the object of meeting the ongoing business needs of the acquirer. In that case the new corporation internally has the task of specifying and developing the system in a manner which is consistent with the objectives of the new organisation. There is in this case no need for a traditional contract to be in place for the development of the system because the new organisation is a single legal entity (consisting of acquirer and developer) and accordingly cannot contract with itself. In such a case therefore the work being performed may not be under a traditional contract which raises the question as to whether estoppel may be applicable in the appropriate circumstances. Relationship models are discussed in more detail in Chapter 5.

The important point to note is that regardless of the current state of any negotiations or contract, equitable estoppel may still be available to prevent unconsionability. It is thus essential to emphasise to the parties involved in any (proposed or actual) information technology contract that in order to avoid the effect of equitable estoppel, neither party must lead the other party (by action or inaction) into believing that a particular state of events exists, to the extent that it constitutes unconscionable conduct. In summary, Bagot\(^{229}\) notes that:

"As a result of Walton v Maher there is no longer any doubt that equitable estoppel is to be regarded as an emanation of equity’s general concern with unconsionable conduct and that it is available to create obligations in situations where representations or promises stopping short of contracts are made. Nor is there any doubt that equitable estoppel is a sword in the plaintiff’s hands. Lawyers advising on the effect of negotiations must now carefully examine the facts to ascertain whether obligations have arisen either in contract or in equitable estoppel. Reliance on the tests of offer and acceptance and consideration are no longer enough. The absence of either may not exclude a party’s liability."

3.2.7.11 The Merger of Equitable and Common Law Estoppel

Although the decision in Walton v Maher\(^{230}\) focused on separate areas of common law and equitable estoppel, more recent cases have merged, to some extent, these doctrines. In Foran v Wight\(^{231}\), Mason CJ found that:

"On further reflection it seems to me that we should now recognise that a common law estoppel may arise out of a representation or mistaken assumption as to future conduct."

---


\(^{230}\) Walton’s (Interstate) v Maher & Anor. (1988) 62 ALJR 110

\(^{231}\) Foran v Wight (1990) 64 ALJR 1
The effect of this decision is that representations as to future conduct, as opposed to representations as to existing fact, may be the subject of common law estoppel and not only equitable estoppel. Mason further elaborated in *Commonwealth v Verwayen*:

"In conformity with the fundamental purpose of all estoppels to afford protection against the detriment which would flow from a party’s change of position if the assumption that led to it were deserted, these developments have brought a greater underlying unity to the various categories of estoppel. Indeed, the consistent trend in the modern decisions points inexorably towards the emergence of one over arching doctrine of estoppel rather than a series of independent rules."

**The Argument Against Estoppel**

The main argument against estoppel generally, whether it be common law or equitable estoppel, is that it threatens to undermine the principals of contract law which have been developed over many years, but more importantly, it may potentially allow a court to set aside a bargain which was agreed between two parties:

"What is clear from the decisions is that in principle the doctrine of estoppel can be applied to pre-contractual negotiations. However it is clear that the circumstances of the negotiation will be examined in detail by a Court. If in general there is equality of the parties and the parties have made conscious decisions not to legally bind themselves based on commercial factors, it may be that a Court will be reluctant to apply the principle of unconscionability in these circumstances. The problems in applying the Walton’s type doctrine of estoppel to pre-contractual negotiations was clearly outlined by Kirby J. In the case of *State Rail Authority of New South Wales v Heath Outdoor Pty. Ltd.* (1986) 7 NSWLR 170 at 176 where he said as follows: ‘Too great a willingness by the Courts to discern, in pre contract negotiations, a basis for estoppel will have the effect of introducing a serious element of uncertainty into our law of contract. It may also encourage expensive litigation in which the terms of the writing are put to one side and the courts busily engaged (as we have been) in a minute examination of the wilderness of pre contract conversations. This may be the reason, at least in the case of written contracts which are accepted by the parties and are not varied or elaborated, to hold the parties to the applicable terms of such contracts and to limit carefully the development of the law of estoppel, least it seriously undermined the adherence to bargains which are such an important feature of modern economic life. But even if estoppel is pushed so, to tender sense of the unconscionable may mislead the courts into substituting the court’s notions of equity as between contracting parties for the bargain which the parties have negotiated and accepted."

Despite this, the strong emphasis on establishing unconscionable behaviour before estoppel will be put into play will hopefully alleviate or minimise the concerns expressed in the above passage.

**Remedies**

Common law remedies and equitable remedies have traditionally be different however with the merging of the doctrines of common law estoppel and equitable estoppel, some consistency in final result should ideally be possible in order to avoid injustice. In *Commonwealth v Verwayen*, Mason C.J. discussed this issue in the following terms:

"So, in Walton’s Stores, a majority of this court concluded that equitable estoppel entitled a party only to that relief which was necessary to prevent unconscionable conduct and to do justice between the parties. Mason C.J. and Wilson J. referred to the statement of Scarman L.J. in *Crabb v Aron District Council* (1976) Ch. 179 at 198, that the court should determine what was ‘the minimum equity to do justice to the plaintiff’. We went on to state:

---

222 *Commonwealth v Verwayen* (1990) 64 ALJR 540 at 545
224 *Commonwealth v Verwayen* (1990) 64 ALJR 540 at 545

Paul P. Parnell

54
‘holding the representor to his representation is merely one way of doing justice between the parties’... it follows, as a matter of principle and authority, equitable estoppel will permit a court to do what is required in order to avoid detriment to the party who has relied on the assumption induced by the party estopped, but no more. In appropriate cases that will require that the party estopped be held to the assumption created, even if that means the effective enforcement of a voluntary promise. To that extent there is an overlap between equitable estoppel generally and estoppel by conduct in its traditional form. But since the function of equitable estoppel has expanded and it has become recognised that an assumption that as to future fact may ground an estoppel by conduct at common law as well as in equity, it is anomalous and potentially unjust to allow the two doctrines to inhabit the same territory yet produce different results.”

Further discussion on specific remedies is provided in later Sections in the context of specific factual situations and case studies.

Conclusion

In summary, the basic principal of estoppel is that “the law should not permit an unjust departure by a party from an assumption which he has caused another party to adopt or accept for the purpose of their legal relations...”\(^{235}\). The practical effect of estoppel is that interaction between parties may give rise to legal rights and obligations regardless of whether a contract exists or not, and regardless of the legal relationship between the parties. Representations of existing fact as well as representations about future conduct may be caught by estoppel.

Because of the developments in the law of estoppel, contracting can no longer be examined without reference to issues raised by estoppel. Interaction between the parties to a large information technology project, for example, can affect the legal rights and obligations of the parties. Elliot summed up the position in relation to estoppel as follows:

> "The doctrine of estoppel as espoused by Walton’s Stores has had a profound effect on the law of contract. Promises not supported by consideration may now be enforceable. Pre-contractual negotiations may become binding. The relief granted pursuant to an estoppel can be equivalent to the relief granted as if a contract existed. Application of the Statute of Frauds and requirements of writing may have no application in the areas of estoppel."\(^{236}\)

Despite this, as actions in estoppel are based on the doctrine of unconscionability, the contractual requirement for consideration has not been entirely "cut up by the roots"\(^{237}\), but rather tempered where unconscionable conduct has occurred\(^{238}\).

Contract law traditionally required a number of elements\(^ {239}\) to be present in a transaction before a binding and enforceable contract came into existence. The law in relation to estoppel has thus opened up the scope of commercial dealings to enable some interaction between parties to give rise to legal rights and obligations outside of the contractual framework. This issue is important because, particularly in large projects, it is likely that many people will be involved in detailed communication before and during and after contracts are in place.


\(^{239}\) that is: offer, acceptance, consideration, capacity, legality and intention to be bound.
Following chapters look at this interaction more closely and the new concept of "micro-agreement" is introduced in an attempt to categorise and describe this interaction more precisely.

### 3.2.8 Collateral Contracts

#### 3.2.8.1 Introduction

In some circumstances a pre-contractual statement may be made by one party to another but that statement is not incorporated into the main body of the contract. Courts have at times found that, when certain pre-requisites are met, that the subject matter of that pre-contractual statement formed a collateral oral contract to the main contract. The end result being that the pre-contractual statement which did not become part of the main contract, was nevertheless given contract weight (including remedies) through a collateral contract.

Under circumstances (described in the following paragraphs) therefore, the legal device of a collateral contract can be used to provide contractual remedies for statements which were made prior to the main contract coming into existence.

In information technology systems integration contracts, quite often a large amount of negotiation between parties will take place prior to a main contract coming into existence. The collateral contract is thus an important area of law, as it may be a means whereby pre-contractual negotiations can be given the force of contract law, including its remedies.

Before collateral contracts can be brought into play however, a number of pre-requisites must be met. These are described in the following paragraphs.

#### 3.2.8.2 Pre-requisites of Collateral Contracts

**3.2.8.2.1 The collateral contract must be consistent with the main contract**

A collateral contract is (usually) an oral agreement which is entered into and supported by the consideration of entering into the main (usually written) contract. The collateral contract must be consistent with the main contract to the extent that the main contract is not interfered with.

The traditional bar of the parol evidence rule to the admission of oral evidence for the purpose of altering or qualifying the main contract is prima facie not an issue. Isaacs J. in *Hoyt's Pty Ltd v Spencer*[^240] addressed the question of the parol evidence in the context of collateral contract. Firstly he explained the concept of parol evidence itself and subsequently went on to describe why it is not such an issue.

> "When two parties are entering into contractual relations with respect to a given subject matter, they may (apart from special technical requirements) elect to conclude their bargain without writing, or they may elect to record it in writing, and, if in writing, they may further decide to have it under seal. But in whatever form they determine to leave their bargain, they may further agree to have one contract only, or to have separate and distinct contracts. All that is for the parties themselves to resolve upon. If they determine to make one contract only, then the terms they decide to include are the only terms that affect them contractually. It connotes that all else is abandoned. And that is the case whatever the form of the contract. If the matter is not committed to writing, though the principal is clear, the evidence is manifestly open to great dispute. But if the parties agree to commit their agreement to writing, then what is written is the conclusive record of the terms of their*

[^240]: *Hoyt's Pty Ltd v Spencer* (1919) 27 C.L.R.133, 143 per Isaacs J.
agreement, and, unless it can be shown that the document was not intended as the complete record of their bargain, no oral evidence can be admitted to alter or qualify it."

Despite the above, Isaacs J. went on to show that, in the context of a collateral contract, the parol evidence rule is not necessarily inconsistent. Specifically, the parol evidence rule cannot have application where the representation made was not intended to "alter or qualify" the main contract. He stated:\footnote{Hoyt's Pty Ltd v Spencer (1919) 27 C.L.R.133, 146 per Isaacs J.}{Hoyt's Pty Ltd v Spencer (1919) 27 C.L.R.133, 146 per Isaacs J.}

"A principle that must govern the bargain of a contractual promise made in consideration of entering into the main contract is that the parties shall have and be subject to all (not some only) of the respective benefits and burdens of the main contract. When the collateral promise is truly consistent with the main contract, that principle has full play. The main contract is not then interfered with. The collateral contract alters, as every contract must, the contractual relations of the parties; but it does not alter, and from the simple statement of the bargain is not intended to alter, the contractual relations which are established by the main contract."

3.2.8.2.2 The consideration for the collateral contract must be the entering into of the main contract

For a collateral contract to exist, there must be a main contract. It is usual that the consideration given for the collateral contract is the entering into of the main contract itself. In other words:

"the consideration given for the promise is no more than the act of entering into the main contract. Going ahead with that bargain is sufficient price for the promise, without which it would not have gone ahead at all."


3.2.8.2.3 The statement must be promissory and not merely representational

The High Court of Australia in \footnote{In J.J.Savage & Sons Pty Ltd v Blakney (1970) 118 C.L.R. 435}{In J.J.Savage & Sons Pty Ltd v Blakney (1970) 118 C.L.R. 435}, considered a situation where the performance of a motor cruiser was the subject of a pre-contractual statement. Specifically, the seller of the motor cruiser stated in a letter to the buyer that the "estimated speed" of the cruiser was 15 m.p.h. Despite this statement by the seller, no subsequent documentation including the contract or specifications for the motor cruiser contained such a statement. The claim by the buyer that a collateral contract existed was rejected by the High Court:

"The actual words used by the appellant in the letter should be considered. So far from being a promissory expression, 'estimated speed 15 m.p.h.' indicates, in our opinion, an expression of opinion as the result 'of approximate calculation based on probability' to use the dictionary equivalent of 'estimate' referred to by the Full Court... the Full Court seems to have thought it sufficient in order to establish a collateral warranty that without the statement as to the estimated speed the contract of purchase would never have been made. But that circumstance is, in our opinion, in itself insufficient to support the conclusion that that a warranty was given. So much can be said of an innocent representation inducing a contract. The question is whether there was a promise by the appellant that the boat would in fact attain the stated speed if powered by the stipulated engine, the entry into the contract to purchase the boat providing the consideration to make the promise effective."\footnote{In J.J.Savage & Sons Pty Ltd v Blakney (1970) 118 C.L.R. 435}{In J.J.Savage & Sons Pty Ltd v Blakney (1970) 118 C.L.R. 435}

3.2.8.3 Collateral contracts and information technology systems integration
It is important to understand that before a collateral contract can be found to exist, a court will require evidence that the statement made was a promise and not merely a representation. In practice, due to the possible difficulty in providing evidence that the statement was a promise, it is obviously preferable to ensure that the subject matter is included specifically in the main contract rather than rely on the device of collateral contracts for relief. Having said that, for various reasons, often requirements which should be specifically in a contract are not. This point was discussed above in Chapter 2 *The Nature of Large Information Technology Projects*, and is also discussed further in the following chapter in Section 0: *Case Study 1 - Micro-agreements in the Context of Requirements Management*

Thus in order to provide sufficient evidence that pre-contractual statements are in fact promissory and not merely representational, a few key points arise from the cases. Firstly, words such as "estimated speed" and "approximate" would *prima facie* tend to indicate a representation rather than a promise. Thus a seller wishing to avoid a collateral contract based on a pre-contractual statement, could use these types of words when describing functionality or performance of the system to be delivered, although of course the statement should not be 'misleading or deceptive' because if so, the *Trade Practices Act, 1974* Section 52 may be invoked. Conversely, a buyer, if intending to rely on such a statement, should either incorporate the statement into the main contract, or have the words included in a formal specification or other document so as to provide better evidence that the statement is a promise. Secondly, the buyer should make it clear that it is prepared to enter into the main contract only on condition that the statement by the seller is a promise and not merely a representation.

Overall, the rule which allows only promises but not representations to become the subject of collateral contracts ensures that formal written agreements freely entered into by the contracting parties carry the force of law and cannot be easily avoided. It is submitted however that under some circumstances this rule may be artificially rigid. As an example, in large software developments it may be very difficult to estimate, at the time of contracting, the actual performance (e.g. in transactions per second) of the system to be eventually delivered. The developer may make statements about such performance but if the system has not yet been built, any such statement will simply be an estimate in most cases. In other words, any statement about system performance could be construed as merely representational and thus, from the buyer's perspective, no remedy would be available through the device of collateral contracts.

### 3.2.8.4 Using collateral contracts to avoid privity of contract

Under some circumstances a promise may be given by a third-party who eventually does not become a party to the main contract. Can that third-party who made the promise be held liable for the consequences if that promise is not met? To illustrate this, as an example a systems integrator may make a promise that a system to be delivered will meet certain performance requirements (e.g. 1000 transactions per second). If the buyer subsequently enters into a contract directly with another third party supplier for the purchase of the system, can the systems integrator be held liable if the claimed performance of that system is not met?

In *Wells (Merstham) Ltd v Buckland Sand And Silica Ltd [1965]* 2 Q.B. 170 it was held that:

- As between A (a potential seller of goods) and B (a potential buyer), two ingredients, and two only, are in my judgement required in order to bring about a collateral contract containing a warranty:
  1. a promise or assertion by A as to the nature, quality or quantity of the goods which B may reasonably regard as being made animo contrahendi; and

Paul P. Parnell 58
Thus it is not necessary that the party who made the promise is a party to the main contract. The party making the promise may still be bound contractually via a collateral contract. Furthermore, the collateral contract may be enforceable notwithstanding that no specific main contract is discussed at the time the promise is given, provided that it "was within the present contemplation of the parties that a contract would shortly be entered into".

From a systems integration point of view, as potentially a large number of parties, contracts, and sub-contracts may be involved, care must be taken to ensure that any promise made can be carried out, whether the promisee actually contracts with the potential buyer or not.

3.2.9 Representations Which Become Terms of the Contract

3.2.9.1 The Issue
Under what circumstances does a statement made prior to the entering into of a contract become a term of that contract?

"When parties enter into a contract, negotiations often precede its execution. A party may make a statement as to some fact, as to some existing or future state of affairs, as to an opinion which he holds, or what he will do in the future. These statements may have the effect, if not necessarily the purpose, of inducing the other party to enter into the contract. But are they to be regarded as terms of the contract?"

3.2.9.2 The Test
So what is the test of whether a statement becomes a term of the contract or not? Denning L.J. in Oscar Chess Ltd. v Williams [1957] 1 W.L.R. 370, stated the position simply as:

"an affirmation at the time of sale is a warranty provided it appears on evidence to be so intended."

In other words, the question of whether a pre-contractual statement becomes a term of the contract simply depends upon the intention of the parties. Denning L.J. qualified his statement above however by saying that firstly, "warranty", in this context, should be given its ordinary English definition, i.e. it means "a binding promise"; and secondly, the intention indicated is based on an objective test, an explanation of which Denning L.J. provided as follows.

"It is sometimes supposed that the tribunal must look into the minds of the parties to see what they themselves intended. That is a mistake. Lord Moulton made it quite clear that 'The intention of the parties can only be deduced from the totality of the evidence.' The question whether a warranty was intended depends on the conduct of the parties, on their words and behaviour, rather than on their thoughts. If an intelligent bystander would reasonably infer that a warranty was intended, that will suffice."
Although the above test of whether a pre-contractual statement becomes a term of the contract appears to relatively straight forward, courts have at times experienced some difficulty in applying it consistently to factual circumstances. As an example, in *Oscar Chess Ltd. v Williams* 250, the buyer of a Morris car was led to believe that it was a 1948 model when in fact it was a 1939 model. Prior to the sale, the seller produced a log book which (unbeknown to both parties) had been fraudulently marked to indicate that the car was made in 1948.

In that case Denning L.J. (in the majority) found that the representation that the vehicle was a 1948 model did not become a term of the subsequent contract. He said:

"...much depends on the precise words that were used. If the seller says 'I believe it is a 1948 Morris. Here is the registration book to prove it,' there is clearly no warranty. It is a statement of belief, not a contractual promise. But if the seller says, 'I guarantee that it is a 1948 Morris. This is borne out by the registration book, but you need not rely solely on that. I give you my guarantee that it is,' there is clearly a warranty. The seller is making himself contractually responsible, even though the registration book is wrong." 251

On the other hand, the dissenting judgement of Morris L.J. found that the representation did become a term of the contract. In making this judgement he took into consideration *Couchman v Hill* [1947] K.B. 554 where Scott L.J. said (at 559):

"as a matter of law, I think every item in a description which constitutes a substantial ingredient in the 'identity' of the thing sold is a condition, although every such condition can be waived by the purchaser, who thereupon becomes entitled to treat it as a warranty and recover damages. I think there was here an unqualified oral condition, the breach of which the plaintiff was entitled to treat as a breach of warranty and recover the damages claimed."

### 3.2.9.3 Practical considerations

The point to note from the above discussion is that, in a practical sense it may be difficult to establish the intention of the parties, and thus whether a pre-contractual statement becomes a term of the contract or not. It is thus important for any parties negotiating a contract to ensure that sufficient and adequate records are kept to provide evidence of intention.

Apart from the above, it has also been noted 252 that the likelihood of a statement being incorporated into a contract can be influenced by a number of factors such as:

(1) The length of time between making the statement and entering into the contract. The longer the time, the more likely it will be seen as a mere representation and not a contractual term. 253

(2) If a written contract which is entered into after the statement is made does not contain that statement, a court may assume, prima facie, that it was not intended to become part of the contract. That is:

"If an oral representation is afterwards recorded in writing, it is good evidence that it was intended as a warranty. If it is not put into writing, it is evidence against a warranty being intended." 254

---

250 *Oscar Chess Ltd. v Williams* [1957] 1 W.L.R. 370
251 *Oscar Chess Ltd. v Williams* [1957] 1 W.L.R. 370 at 375 per Denning L.J.
252 *e.g.* see Khoury, D. and Yamouri, Y.S. "Understanding Contract Law", Butterworths, 1985 at 69.
253 *e.g.* see: *Oscar Chess Ltd. v Williams* [1957] 1 W.L.R. 370 at 373 per Denning L.J.
254 *Oscar Chess Ltd. v Williams* [1957] 1 W.L.R. 370 at 376 per Denning L.J.
(3) If a party has special knowledge or skill then a court will assume that he is capable of using that knowledge or skill to ascertain the accuracy or otherwise of the pre-contractual statement. If that skill or knowledge is not used then the court will be less likely to find that the statement forms part of the contract.

(4) If the statement is a matter of importance to the parties then a court is more likely to find that it is a term of the contract rather than a mere representation. During contract negotiations therefore parties need to ensure that issues of importance are clearly made known to the other party.

3.2.10 Good Faith and Reasonableness in Contractual Situations

In *Service Station Association Limited v Berg Bennett & Associates Pty Ltd* 256 it was found that Australian law does not necessarily imply obligations of reasonableness or good faith into contracts. Despite this however, the *Trade Practices Act, 1974 Part VI* provides remedies for misleading and deceptive conduct and as such obligations related to honest dealings are present via non-contractual means257. In addition, it may be possible to imply such terms if the general tests applicable to implied terms are met - these tests are described below in "Implied Contractual Terms".

3.3 EXPRESS CONTRACTUAL TERMS

3.3.1 Introduction

Once a contract exists, the terms of the agreement may be embodied in a written document, an oral contract, or indeed it may be partly oral and partly written. In addition, some of the terms may be express and others implied. For various reasons, courts are often called upon to interpret contracts and to establish the intention of the parties when entering into such contracts.

Accordingly, a number of legal rules have developed over time to resolve such problems as uncertainty and ambiguity of the terms of a contract. It is the intention of this Section to identify these rules in a general sense and, where applicable, discuss particular issues relating to information technology contracts.

3.3.2 Contracts in Writing - Admissibility of Extrinsic Evidence

3.3.2.1 The Parol Evidence Rule

3.3.2.1.1 The Rule

The parol evidence rule states that where a contract is wholly in writing, then no evidence is admissible which would add to, vary or contradict the written document. In other words:

"where a contract is reduced into writing, where the contract appears in the writing to be entire, it is presumed that the writing contains all the terms of it, and evidence will not be admitted of any previous or contemporaneous agreement which would have the effect of adding to or varying it in any way."258

---

255 e.g. see: *Oscar Chess Ltd. v Williams* [1957] 1 W.L.R. 370 at 376,377 per Denning L.J.
256 *Service Station Association Limited v Berg Bennett & Associates Pty Ltd* (1993) 45 FCR 84, 27 IPR 23
258 *Merchantile Bank of Sydney v Taylor* (1891) 12 LR (NSW) 252, at 262 per Innes J., as quoted in Khoury,
But how do you tell if a contract is wholly in writing? This needs to be established before the parol evidence rule can be put into play. Hope J.A. stated the position as follows:

"...the correct rule is that the existence of writing which appears to represent a written contract between the parties is no more than an evidentiary foundation for a conclusion that their agreement is wholly in writing. The tendering of evidence, whether oral or in writing, to prove a contractual term cannot be excluded until it is determined that any term in writing recorded the whole of the party's agreement."\textsuperscript{259}

The above point should be noted when examining "Entire Agreement" clauses in contracts - prima facie they provide evidence of the party's intention to have the agreement wholly contained in the written document, but other evidence may be available and admissible to show otherwise.

Once it has been established that an agreement is entirely in writing, it is "wrong to consider extraneous circumstances... and [the] document should be considered according to its contents alone and no reference to background information should be made."\textsuperscript{260}

3.3.2.1.2 Exceptions to the Parol Evidence Rule

In order to avoid injustice by strict application of the parol evidence rule, a number of exceptions to the rule have been developed. The main ones are described in the following paragraphs.

(1) \textit{Extrinsic oral evidence may be admissible as an aid to clarifying ambiguities in the written contract.}

In a technical sense, there are two types of ambiguity - patent ambiguity and latent ambiguity. Patent ambiguity occurs when there is an ambiguity in the document itself. Latent ambiguity occurs when there is (for example) a reference to two or more persons or objects which have the same name, even though the document itself is unambiguous.

In the case of patent ambiguity, evidence may be admissible to clarify the parties or subject matter of the contract, or to interpret words in a foreign language or words which are used in a technical sense. A number of cases have addressed these issues although an important point to bear in mind is that:

"As in relation to the question of identifying the parties, there is ample authority to suggest that the basis of the admission of the evidence as to the subject matter is not the fact that the document is ambiguous, but the fact that the court is entitled to be placed in the same position as the parties in order to be able to interpret the instrument they have executed."\textsuperscript{261}

In relation to latent ambiguity, it has been said that:

"The words of the instrument may be perfectly plain and unambiguous... but if, from the surrounding circumstances, when you come to apply the instrument, you find that there are

---

\textsuperscript{259} Norwest Beef Industries Ltd. v Peninsula and Oriental Steam Navigation Co. (1987) 8 N.S.W.L.R. 568 at 570, per Hope J.A., quoting State Rail Authority of New South Wales v Heath Outdoor Pty. Ltd. (1986) 7 N.S.W.L.R. 170 per McHugh J.A.

\textsuperscript{260} British Movietone News Ltd. v London & District Cinemas Ltd (1952) AC 166, on appeal to the House of Lords, as quoted in Khoury, D. and Yamouni, Y.S. "Understanding Contract Law", Butterworths, 1985.

two persons who satisfy the words ... or two markets to which the parties have been referring, there is a latent ambiguity, and evidence is admissible to resolve it. 262

It should be noted however that the purpose of admitting the evidence is:

"not to contradict or vary the contract, but to apply it to the facts which the parties had in their minds and were negotiating about." 263

(2) If a written document requires rectification because of a mistake by the parties, a court may under some circumstances admit extrinsic evidence for that purpose of rectifying that mistake.

Specifically,

"[The question is] whether it was the common intention of the parties in signing the agreement to record the oral agreement which they had already concluded or whether, on the other hand, they intended to make a new and different contract by the writing which they signed." 264

(3) If it was the intention of the parties that general custom or trade usage terms were to be part of the agreement, but were not specifically incorporated into the written contract, then a court may admit extrinsic evidence to show such usage.

A statement of the law in this regard is as follows:

"It has long been settled that in commercial transactions extrinsic evidence of custom or trade usage is admissible to annex incidents to written contracts in matters with respect to which they are silent... this has been done on the principle or presumption that, in such transactions, the parties did not mean to express in writing the whole of the contract by which they intended to be bound, but a contract with reference to those known usages." 265

Despite the above however it is important to note that:

"It must be so notorious that everybody in the trade enters into a contract with that usage as an implied term. It must be uniform as well as reasonable, and it must have quite as much certainty as the written contract itself." 266

(4) If a contract was intended to come into operation after a certain event takes place, then oral evidence may be admitted to show that. 267

Other issues arise in the context of the parol evidence rule which relate to contracts containing technical language. In such cases it has been held that extrinsic evidence may be admissible to resolve any ambiguities. Obviously high technology developments (including information technology projects) involve a significant degree of technical language. The scope of the parol evidence rule should be understood in this context. Some commentators have indicated that this "technical language" exception to the rule is not necessarily an exception. For example:

"If the admissibility of evidence is to be tested against the need to give effect to the parties' intentions, there is no reason for regarding the technical meaning category as an exception

262 Great Western Ry v Bristol Corp. (1918) 87 L.J. Ch. 414 at 429 per Lord Wrenbury.
264 Maralinga Pty Ltd v Major Enterprises Pty Ltd (1973) 128 CLR 336, at 343 per Barwick C.J.
265 Hutton v Warren [1936] 1 M & W 466, at 475 per Baron Parke.
266 Nelson v Dahl (1879) 12 Ch.D 558, at 575 per Jessel MR.
267 e.g. see: Pym v Campbell (1856) 6 E&B 370
3.4 IMPLIED CONTRACTUAL TERMS

3.4.1 Introduction

Although the parties to a contract may not explicitly agree to particular terms, they may nonetheless be bound in certain circumstances by implied terms. Implied terms are "in effect, obligations imposed on the parties by the law in the absence of contrary agreement." Overall, the concept of implied contractual terms is intended to "promote fair and reasonable contract performance," and is "designed to give effect to the parties' presumed intention" (even though that intention was not explicitly stated).

There are a number of situations in which the law will allow the implication of terms and there have been various attempts at categorising these. For the purposes of this document, firstly a discussion will be presented on the general tests to be applied in relation to the incorporation of implied terms; secondly, general categories will be discussed including the effect of custom, prior transactions, statute etc.; and thirdly, where applicable a specific focus on information technology systems integration contracts will be made.

3.4.2 The Test for Implied Terms

In *B.P. Refinery (Westernport) Pty Ltd v Shire of Hastings*, the Privy Council held that, in relation to contracts which are written and which purport to be formal and complete documents:

"For a term to be implied, the following conditions (which may overlap) must be satisfied:

1. it must be reasonable and equitable;
2. it must be necessary to give business efficacy to the contract, so that no term will be implied if the contract is effective without it;
3. it must be so obvious that 'it goes without saying';
4. it must be capable of clear expression;
5. it must not contradict any express term of the contract."

This passage has been accepted and applied on many occasions by the High Court and thus carries significant weight.

It is important to note that the above test does not apply if the contract is not recorded in writing in a formal sense. In such a case the test as to whether a term may be implied into

---

270 Renard Constructions (ME) Pty Ltd v Lachlan Resources NL (1988) 26 NSWLR 234 at 271 per Priestly J.
271 Codella Construction Pty Ltd v State Rail Authority of New South Wales (1982) 56 A.L.J.R. 459 at 461 per Mason J.
273 B.P. Refinery (Westernport) Pty Ltd v Shire of Hastings (1977) 16 ALR 363 at 365
274 for example: Codella Construction Pty Ltd v State Rail Authority of NSW (1982) 149 CLR 337 at 347, 404; (1982) 55 A.L.J.R. 459 at 461 per Mason J; *Hospital Products Ltd v United States Surgical Corporation* (1984) 156 CLR 41 at 65-6, 95, 117-8, 121; *Byrne v Australian Airlines Ltd* (1995) 131 ALR 422 at 428, 443

---

Paul P. Parnell

64
a contract is whether the implication of the term is “necessary for the reasonable and effective operation of a contract of that nature in the circumstances of the case”\textsuperscript{276}.

### 3.4.2.1 The Implied Term Must be Reasonable and Equitable

**Reasonable**

It seems clear that unless a term is reasonable it will not be implied by a court. As an example, in the *B.P. Refinery* case it was stated that:

> "the alleged implied term is to be rejected on a simpler and clearer ground that, taking into account the matrix of facts in which that agreement was set, to imply such a term would be wholly unreasonable and inequitable"\textsuperscript{276}.

In this case, the Court felt that the proposed implied term was unreasonable because, for the term to be implied, B.P. would have had to make massive corporate structural changes in order to meet its obligations under that implied term.

Other examples of where the concept of reasonableness has been discussed include *Liverpool City Council v Irwin* where it was held:

> "... unless a warranty or term is in all the circumstances reasonable there can be no question of implying it into a contract..."\textsuperscript{277}.

And *Young and Marten Ltd. v McManus Childs Ltd.:

> "No warranty ought to be implied in a contract unless it is all the circumstances reasonable"\textsuperscript{276}.

**...and Equitable**

In a sense, the requirement that an implied term be “equitable” indicates that that term must be reasonable or fair\textsuperscript{279} from the point of view of both parties\textsuperscript{280}, although in practice the terminology used is that of the “officious bystander”\textsuperscript{281}. This concept is discussed further below under the heading *The Implied Term Must be Obvious*.

### 3.4.2.2 The Implied Term Must be Necessary to Give Business Efficacy to the Contract

**Necessary**

Failure to meet this requirement is one of the most common reasons why courts reject proposed implied terms\textsuperscript{282}. The difficulty arises primarily in relation to showing that the term is “necessary”.

In *Re Comptoire Commercial Anversois and Power Son and Co.*\textsuperscript{283} the requirement for necessity was phrased as follows:

\textsuperscript{276} Hawkins v Clayton (1988) 164 CLR 539 at 573
\textsuperscript{276} B.P. Refinery (Westemport) Pty Ltd v Shire of Hastings (1977) 52 ALJR 20 at 27
\textsuperscript{277} Liverpool City Council v Irwin [1977] A.C. 239 at 262 per Lord Salmon.
\textsuperscript{278} Young and Marten Ltd. v McManus Childs Ltd. [1969] 1 A.C. 454 at 465 per Lord Reid.
\textsuperscript{281} for example see: Codeifa Construction Pty Ltd v State Rail Authority of NSW (1982) 56 A.L.J.R. 459 at 473 per Aickin J.
\textsuperscript{283} Re Comptoire Commercial Anversois and Power Son and Co. [1920] 1 K.B. 868 per Scrutton L.J.

Paul P. Parnell

65
"it must be such a term that both parties must have intended that it should be a term of the contract, and have only not expressed it because its necessity was so obvious that it was taken for granted".

In relation to the question as to how strictly a court will apply this requirement, it is clear that a term will not be implied into a contract if it just reasonable to do so. More is required - it must be necessary to the extent that:

"such obligation should be read into the contract as the nature of the contract itself implicitly requires, no more, no less: a test, in other words, of necessity." 264.

Despite this however it is likely that a term need not be so necessary that the entire contract cannot be performed without it265.

...for Business Efficacy

The necessity of the term must be in relation to the business environment. The concept of necessity, by itself, is therefore insufficient unless it is linked to business efficacy. In The Moorcock, 266 Bowen L.J. said that:

"In business transactions such as this, what the law desires to effect by the implication is to give such business efficacy to the transaction as much have been intended at all events by both parties who are businessmen...".

Examples of cases where a term was implied include Fraser v Thames Television Ltd.287 (where a term was implied into a contract that a television company would not use an idea for a television programme, unless it offered the originators of the idea acting roles in the series); and Metropolitan Electric Supply Co. Ltd. v Ginder268.

As an analogy to Fraser v Thames Television Ltd., it may be possible to argue (for example) that it was an implied term of a contract that a systems integration company would not use the idea for a new software product unless the originators of that idea were engaged to develop that product. In some ways it is submitted that this concept provides a level of intellectual property protection with contractual force, which is separate from the traditional intellectual property protection mechanisms such as copyright, patents etc.

Examples of cases where a term was not implied into a contract can be found in Esso Australia Resources Ltd. v Plowman288 (term relating to confidentiality in an arbitration agreement); and Hospital Products v United States Surgical Corporation290 (a term requiring the distributor of products to refrain from adversely affecting the market for USSC products was not implied as it was not necessary for USSC’s commercial objective, and as such, the contract was not unworkable291 without the implication of the term).

In the context of large information technology projects, for example, it may be an implied term of a contract that a contractor will use reasonable skill and care and perform the work, or to perform it in a "workman-like" manner (see: Zorba Structural Steel Company

264 Liverpool City Council v Irwin [1977] A.C. 239 per Lord Wilberforce. See also: Codelfa Construction Pty Ltd v State Rail Authority of NSW (1982) 149 CLR 337 at 346 per Mason J.
266 Renard Constructions (ME) Pty Ltd v Minister for Public Works (1992) 26 NSWLR 234 at 257-8.
267 The Moorcock [1989] 14 P.D. 64, Bowen L.J.
268 Fraser v Thames Television Ltd. [1984] Q.B. 44.
269 Metropolitan Electric Supply Co. Ltd. v Ginder [1901] 2 Ch. 799. See also a number of other examples in Lewison K. Interpretation of Contracts Sweet & Maxwell, London, 1989 at 102-3.
287 Esso Australia Resources Ltd. v Plowman (1995) 131 ALR 422.
290 Hospital Products v United States Surgical Corporation (1984) 156 CLR 41.
291 Hospital Products v United States Surgical Corporation (1984) 156 CLR 41 at 66.
Pty Ltd v Walco"292); or that the parties will co-operate to ensure the contract is performed effectively293. Due to the nature of large information technology projects it is likely that a degree of uncertainty exists and that many decisions will need to be made on issues not fully clarified before entering into the main contract. Section 2.5 above discussed how requirements can change throughout the life of a large project and this is one area where co-operation between the parties could be beneficial in resolving issues. The implication of such terms is likely to be beneficial in performing the contract and reaching the desired outcomes of each of the parties.

3.4.2.3 The Implied Term Must be Obvious

In establishing what is obvious, Courts referred to what the parties "would most likely have agreed had they considered the point"294. In order to do this the officious bystander test is applied. In summary, this test has been stated as follows:

"Prima facie that which in any contract is left to be implied and need not be expressed is something so obvious it goes without saying; so that if, while the parties were making their bargain, an officious bystander were to suggest some express provision for it in their agreement, they would testify suppress him with a common 'Of course'"295.

Care must however be taken in applying the "officious bystander" concept because, as explained by Aicken in Codelfa Construction Pty Ltd v State Rail Authority of NSW:

"The first problem is that the manner in which the officious bystander formulates his question will often determine the answer which the parties will give."296

It should also be noted that if more than one possible formulation of an implied term could be used, it may be difficult to claim that any one is obvious:

"No covenant ought ever to be implied unless there is such a necessary implication that the court can have no doubt what covenant or undertaking they ought to write into the agreement."297.

It is submitted that a failure to properly assess risk will, in itself, be an insufficient reason to claim that a term is obvious for the purpose of implication into a contract. In the Codelfa Construction case, a common assumption was made which was incorrect and because of which no provision was made in the contract to cover a particular risk. In the words of the court:

"This is not a case in which an obvious provision was overlooked. Rather the parties made a common assumption which masked the need to explore what provision should be made to cover the event which occurred. Negotiation might have yielded any one of a number of alternative provisions, each being reasonable."298.

From a practical point of view, failure to properly assess risk in information technology contracts is likely to be, in itself, insufficient reason to claim the incorporation of an implied term to overcome that risk.

292 Zorba Structural Steel Company Pty Ltd v Walco (1993) 115 FLR 206
293 RDJ International Pty Ltd v Performed Line Products (Australia) Pty Ltd (1996) 39 NSWLR 417
295 Shirlaw v Southern Foundaries (1926) Ltd [1939] 2 K.B. 206 per Mackinnon L.J.
296 Codelfa Construction Pty Ltd v State Rail Authority of NSW (1982) 56 A.L.J.R. 459 at 473 per Aickin J.; For an example where the Court framed a question to the officious bystander in such a way as to be possibly suggestive of only one answer, see: Spring v National Amalgamated Stevedores and Docker Society [1956] 1 W.L.R. 585.
298 Codelfa Construction Pty Ltd v State Rail Authority of NSW (1982) 149 CLR 337 at 355 per Mason J.
3.4.2.4 The Implied Term Must be Capable of Clear Expression

Although this requirement appears to be obvious, the important point to note is that although it may be possible to phrase a term using normal English language constructs, the wording must be precise enough to be applied in the legal context. For example:

"The second obstacle is the difficulty in formulating the proposed terms, as demonstrated by the vagueness and the ambiguity inherent in such words as 'discriminate' and 'abnormality'."

The High Court in Ansett Transport Industries v Commonwealth[^300] has also indicated that width and lack of precision of an implied term will prevent its implication. Despite this however, imprecise wording does not appear to be an absolute bar to the implication of a term, but rather, such a characteristic would make it less likely for a court to accept it[^301].

3.4.2.5 The Implied Term Must Not Contradict an Express Term

The basis for this requirement is to not go against the presumed intention of the parties evidenced by the written agreement. The principle has been judicially stated as follows:

"It is, of course, impossible to imply in a contract any term or condition inconsistent with its express conditions, or with the intention of the parties as gathered from those provisions. The first thing, therefore, in every case is to compare the term or condition which is sought to imply with the express provisions of the contract, and with the intention of the parties as gathered from those provisions, and ascertain whether there is any inconsistency."[^302]

3.4.2.6 Implied Terms and Information Technology Contracts

In large information technology contracts that involve many organisations and individuals developing components, there is likely to be a large amount of change and uncertainty involved in the technical aspects of the work, particularly during the early stages. Because of this is it likely that many issues will arise in the context of such projects that may lead to claims for the implication of terms into such contracts. As an example, it is possible (and in the author's experience quite common) for technical performance issues to not be fully addressed in a contract, despite the obvious importance of such terms. Specifically, issues such as how fast the system operates, how many transactions per second the system can process, the rate of screen updates and similar issues may be critical to the effective operation of the system to be delivered but may not be fully specified at the time of contract.

Can a term related to system performance be implied into a contract? Based on the test in B.P. Refinery (Westemport) Pty Ltd v Shire of Hastings[^303], it will need to be shown that the term to be implied:

(a) *is reasonable and equitable*: Depending upon the particular factual situation it is likely that a term implying system performance could be introduced into a contract. However, the degree of reasonableness and equity is likely to revolve around the specific performance levels to be implied. As an example, the buyer of the system may wish a screen update rate of 0.1 seconds when a "reasonable" figure for similar systems may be a range of (e.g.) 1 to 2 seconds.

[^300]: Ansett Transport Industries v Commonwealth (1977) 139 CLR 54 per Gibbs J.
[^303]: B.P. Refinery (Westemport) Pty Ltd v Shire of Hastings (1977) 16 ALR 363 at 365
(b) *is necessary to give business efficacy to the contract.* It is conceivable that a term implying the performance of an information technology system could be necessary to give business efficacy to the contract. A contract for a system that gives inadequate performance to the extent that that system is unable to be used for its intended purpose is unlikely to be entered into by the buyer if it knows such a problem will occur. Despite this, the degree of necessity could be determined, once again, by the performance figures to be implied. Using the example in (a) above, a term implying 0.1 second screen update rate may not be 'necessary' although a term implying a 2 second update rate may be (depending on the context).

(c) *is so obvious that it goes without saying:* Depending on the context, it is likely to be obvious that a computer system needs to perform at a level adequate for its intended purpose.

(d) *is capable of clear expression:* Specifications relating to performance are often used in technical documentation and in most cases can be stated in clear and measurable terms. A simple example is that: "

“The System must display results of a Transaction to the User within 1.0 second plus or minus 0.1 seconds from the time that that User enters the Process command."

(e) *does not contradict any express term of the contract.* Many information technology contracts do not identify performance explicitly (although often refer to a schedule to the contract containing a specification if available at the time).

Based on the above points it is conceivable that system performance levels could be implied into a contract, although naturally the particular factual situation needs to be taken into account to determine this on a case by case basis. This discussion provided one example of how a term may be implied. The following chapter discusses a number of key activities associated with large system development and it is submitted that, depending upon the circumstances, terms relating to those activities may be implied.

### 3.4.3 Terms implied by custom

#### 3.4.3.1 The Rule

In *Con-Stan Industries of Australia Pty. Ltd. v Norwich Winterthur Insurance (Australia) Limited*\(^\text{304}\), the High Court summarised the elements required to allow the implication of terms into a contract on the basis of custom. Specifically, the Court expounded the following propositions:

"(1) The existence of a custom or usage that will justify the implication of a term into a contract is a question of fact (Nelson v Dahl (1879) 12 Ch.D. 568, 575).

(2) There must be evidence that the custom relied on is so well known and acquiesced in that everyone making a contract in that situation can reasonably be presumed to have imported that term into the contract. ... However, it is not necessary that the custom be universally accepted, for such a requirement would always be universally defeated by the denial by one litigant of the very matter that the other party seeks to prove in the proceedings."

(3) A term will not be implied into a contract on the basis of custom where it is contrary to the express terms of the agreement.

(4) A person may be bound by a custom notwithstanding the fact that he had no knowledge of it.

Apart from the above, if the custom itself has some independent method of enforcement, it will not be implied into a contract: *Byrne v Australian Airlines Ltd.* (1995) 131 ALR 422.

### 3.4.3.2 Customs and Information Technology

"Custom" has been defined as "the usual way of acting in given circumstances." From the point of view of information technology contracts, it is submitted that customs exist which could, in the appropriate factual situation, be effectively incorporated into a contract as implied terms. The following chapter "Micro-agreements" will undertake a comprehensive analysis of some of the major aspects of software development and acquisition activities, and as such should identify the usual "way of acting in given circumstances" so as to help identify information technology custom.

Some brief examples of the types of activities which may, under appropriate circumstances, be given the status of "custom" for the purpose of implying a term into a contract include:

- It may be an implied term of a information technology contract that a *Requirements Analysis* phase is conducted before major effort is expended on other aspects of a project;
- It may be an implied term that sufficient *configuration management* processes are in place to ensure that at any time in the development, changes can be adequately tracked;
- It may be an implied term that before a software sub-system is provided to the system integrator for integration into the larger system, that a formal process of validation and quality assurance was in place.

Further discussion and examples will be provided in a following chapter. For the present purposes however it is important to note that the mechanism of implying contractual terms on the basis of custom may be a powerful way of ensuring contractual weight is available.

### 3.4.4 Terms derived from prior transactions

It is possible for a term to be implied into a contract on the basis of a prior dealing. In summary, the requirements for this to occur are:

- (a) there has to be a means of identifying the terms previously employed, which will in most cases involve reference to the use of a contractual document on these occasions;
- (b) those previous occasions must be sufficiently numerous and frequent;
- (c) and the conduct consistent enough, to constitute a regular course of dealing,
- (d) which raises the reasonable expectation that the same terms should be included in the subsequent contract.

---

306 For example, this basis for implying terms is mentioned in: *Breen v Williams* (1996) 186 CLR 71 at 79, 91; *Esso Australia Resources Ltd v Plowman* (1995) 69 ALJR 404.
In practice the above requirements can be onerous to meet and in particular situations the implication of such terms has been specifically rejected by the courts. For example, implication has been rejected where the prior terms were not clearly contractual\(^{308}\), and where insufficient notice of the prior term was given, particularly in relation to excluding or limiting terms\(^{309}\).

Overall, when considering whether terms can be implied into a contract on the basis of prior dealing it is important to consider the actual intention of the parties to that contract\(^{310}\). One commentator has stated that:

\[\text{"On the whole, it seems unwise to elevate a recognition that parties who have previously made similar contracts with each other may intend the same terms to apply again to the status of a presumption justifying implication of those terms."}\]

This comment is particularly relevant to information technology contracts where the rate of change of technology may make particular techniques and methods obsolete in relatively short timeframes. As an example, a few years ago design methods such as "functional decomposition\(^{312}\) (or standards requiring such methods\(^{313}\)) may have been prescribed. Today, object oriented\(^{314}\) methods are more likely to be required. Internet technologies are another area where there may be some danger in implying terms from prior dealing. For example, even ten years ago "publication" of material on-line on a world wide web page would not have been possible, while today that practice is common place.

Despite these dangers there is obviously some value to the implication of terms based on prior dealing provided there is a structured assessment of whether the term should be implied. The requirements for implying such terms discussed above, it is submitted, are adequate to cope with technological change although much care needs to be taken when doing so.

3.4.5 Terms applicable to contracts of a particular class

Provided that there is no inconsistency with the express terms of a contract\(^{315}\), if a particular class of contract exists\(^{316}\) then it may be possible to imply or inherit terms from that class into specific contracts.

Classes of contract in which terms have been implied include for example: “sale of goods, supply of labour and materials, sale and lease of land, mortgage, guarantee, insurance, employment, agency, partnership”\(^{317}\).

\(^{308}\) Eggleson v Marley Engineers Pty Ltd (1979) 21 SASR 51; D.J. Hill & Co. Pty Ltd v [1971] VR 749.
\(^{309}\) e.g. Thomas National Transport (Melbourne) Pty. Ltd. v May & Baker (Australia) Pty. Ltd. (1966) 115 CLR 353; Sydney City Council v West (1955) 114 CLR 481.
\(^{310}\) e.g. Quinn v Jack Chia (Australia) Ltd [1992] 1 VR 587.
\(^{312}\) “functional decomposition” is a method of designing software by breaking it into separate modules, each of which performs a particular function.
\(^{313}\) such as the United States military standard MIL-STD-1679A.
\(^{314}\) object-oriented systems are broken down into components that aim to represent real-world ‘objects’ or classes of objects, with each having attributes and services (stated simplistically). As an example, a Ford Falcon may be represented as an object of the class motor vehicle and have attributes such as colour, number of wheels etc.
\(^{316}\) e.g. Bryne v Australian Airlines Ltd (1995) 131 ALR 422 at 426; Breen v Williams (1996) 186 CLR 71 at 103.
The categories of class are not closed\(^{318}\) although the current standards are applicable\(^{319}\). Some cases which consider new classes of conduct include: Hospital Products Ltd v United States Surgical Corporation\(^{320}\) (distributorship agreement); and Esso Australia Resources Ltd v Plowman\(^{321}\) (arbitration agreement).

The question of whether a special class of information technology systems integration contracts exists, or is required, will be considered in more depth in Chapter 3 "Micro-agreements". In summary, that chapter describes how large information technology projects are usually undertaken with a number of "key practices" in place. These include requirements management, quality assurance, project planning, project tracking and oversight, subcontractor management and configuration management.

In light of the detailed discussions provided in chapter 3, it is submitted that a new "class" of contract could be established for large scale information technology contracts, where terms relating to these "key practices" could be implied. Any such implication should however be in line with the general tests for the implication of such terms discussed earlier (section 3.4.2).

### 3.4.6 Time for Performance

In many agreements, particularly those which involve large complex development processes, research and development, or high risk activities generally, the time required to perform the work can be difficult to estimate.

"When the language of a contract does not expressly, or by necessary implication, fix any time for the performance of a contractual obligation, the law implies that it shall be performed within a reasonable time. The rule is general application, and is not confined to contracts for the carriage of goods by sea. In the case of other contracts the condition of reasonable time has been frequently interpreted; and has invariably been held to mean that the party upon whom it is incumbent duly fulfils his obligation, notwithstanding protracted delay, so long as such delay is attributable to causes beyond his control, and he has neither acted negligently nor unreasonably.\(^{322}\)"

In relation to information technology systems integration contracts the above can be of particular importance because due to the nature of systems integration projects, late delivery of one of the components of a system from a supplier can have follow-on effects for the systems integrator in terms of schedule. A large complex development may involve the systems integrator bringing together components of a system from a "hierarchy" of suppliers. Contracts dealing with such developments are likely to provide definitive schedules for delivery of the final fully-integrated system, but may not necessarily specify the time for performance of some of the less significant or intermediate contract requirements.

In order to establish a "reasonable time for performance", presumably data must be gathered from the industry, or expert evidence must be called. It is submitted that the concept of "reasonable time" may be difficult to establish in the context of large systems integration contracts because firstly, the number of very large projects is relatively few; secondly, published data on actual time for performance may be difficult and costly to obtain due to the proprietary nature of such information; and thirdly, the time for performance of contractual obligations in a systems integration contract will be somewhat driven by the technology of the day. By the time that the project is complete and time-for-

---

\(^{318}\) Castlemaine Toeyes Ltd v Carlton & United Brewers Ltd. (1987) 10 NSWLR 468 at 487 per Hope JA.

\(^{319}\) Renard Constructions (ME) Pty Ltd v Minister for Public Works (1992) 26 NSWLR 234 at 261.

\(^{320}\) Hospital Products Ltd v United States Surgical Corporation (1984) 156 CLR 41 at 62

\(^{321}\) Esso Australia Resources Ltd v Plowman (1995) 69 ALJR 404

\(^{322}\) Hick v Raymond & Reid [1893] AC 22 at 32 per Lord Watson.
performance data is available, technology has more than likely moved on to such an extent that that data is likely to be somewhat invalid.

3.4.7 Implying Terms as to the Duration of a Contract

As a separate issue to the time of performance discussed above, (but one is still related to time), if there is no express term in a contract which provides for the duration of the agreement, it may be possible to imply that the duration of that agreement is a reasonable time. In practice this results in an implied term giving the right to terminate upon reasonable notice: Crawford Fitting Co v Valve & Fittings Pty Limited. In relation to reasonable notice it has been said that:

“It is true that it does not require very much to induce a court to read into an agreement of a commercial character, either by construction or by implication, a provision that the arrangements between the parties, whatever they may be shall be terminable only upon reasonable notice.”

Overall, however the situation is dependant upon the circumstances of each case, and the intention of the parties:

“An agreement which is silent about determination will not be determinable unless the facts of the case, such as the subject matter of the agreement, the nature of the contract or the circumstances in which the agreement was made, support a finding that the parties intended it to be determinable, but there is in my judgement, no presumption one way or the other.”

In large information technology projects the duration of a contract may become an issue in respect of some ongoing work. Maintenance contracts, for example, are often put in place for software systems to ensure that the users ultimately have the benefit of ‘bug’ fixes as well as updates and new releases to the software that are produced in response to changing business needs and user requirements.

These contracts are usually put in place for an initial fixed period (e.g. 12 months) and can (subject to the wording of the contract) be extended automatically unless terminated upon notice by either party. Where an obligation is continuing but there is no express term defining the duration of the contract, then that duration may be implied as a reasonable time. What then is a reasonable time? Naturally it will depend upon the particular factual situation but it is considered that a “reasonable” time will usually be longer for large complex projects than for smaller ones. In a very large and complex project, usually both parties will benefit by having (say) a 12 month or longer maintenance contract in place. The buyer will obtain the benefits of maintenance while the seller will have a degree of certainty to enable it to employ staff and engage other resources. It is conceivable that a reasonable time for large projects therefore could easily be 12 months or more.

Despite this it should be noted that termination provisions are also quite common and provide, for example, that after an initial term, either party may terminate upon (say) 90 days notice. Implying a term relating to the duration of a contract may need to be done in the context of relevant termination provisions also.

3.4.8 Implying Terms Related to Price

323 Crawford Fitting Co v Valve & Fittings Pty Limited (1988) 14 NSWLR 438
324 Australia Blue Metal Ltd. v Hughes [1963] A.C. 74 (Privy Council)
325 Re Spenborough Urban District Council’s Agreement [1968] Ch. 139, per Buckley J.
In general, where a contract contains no express statement regarding price, then often it may be implied into that contract that the price is to be a reasonable price: e.g. Foley v Classique Coaches Ltd.\textsuperscript{326} In the Foley case, the contract entered into by the parties was “at a price to be agreed”. As there was in fact a contract in existence, the court was able to imply a term into that contract that the price was to be a reasonable price.

Despite the above, if no contract is yet in place, then naturally a court cannot imply such a term unless it was clearly the intention of the parties to do so:

“The first question must always be whether any legally binding contract has been made, for until that issue is decided a court cannot properly decide what extra terms, if any, must be implied into what ex hypothesi is a legally binding bargain, as being both necessary and reasonable to make that legally binding bargain work. It is not correct in principle, in order to determine whether there is a legally binding bargain, to add to those terms which alone the parties have expressed, further implied terms upon which they have not expressly agreed and then by adding the express terms and the implied terms together thereby create what would not otherwise be a legally binding bargain.”\textsuperscript{327}

This issue is particularly important in the context of micro-agreements. When a micro-agreement is formed by parties to a large information technology project, additional work may be necessary as a result of the micro-agreement, even though that work was not envisaged under the main contract. The implication of a price for the additional work in such a situation may be critical to the parties especially if profit margins are already narrow. As we will see in the following chapters micro-agreements can be common in large information technology projects and accordingly if express terms relating to price are not put in place, the implication of such terms may be important.

### 3.4.9 Implied Terms of the Standard of Performance

At Common Law, there is usually an implied term that goods provided under a contract will be reasonably fit for the purpose intended. Specifically, it has been stated that:

“I do not hesitate to say that I am clearly of opinion, as a general proposition of law, that when one man engages with another to supply him with a particular article or thing, to be applied to a certain use or purpose, in consideration of a pecuniary payment, he enters into an implied contract that the article or thing shall be reasonably fit for the purpose for which it is to be used and to which it is to be applied.”\textsuperscript{328}

Likewise if a contract is for the supply of services, there is an implied condition that the services will be provided in a workmanlike manner, and that materials used will be good quality.\textsuperscript{329}

Implied terms relating to standard of performance may be implied through Common Law means however modern Australian laws generally provide legislative mechanisms for the implication of such terms. Accordingly, it is likely that legislation such as the Trade Practices Act 1975, or state Sales of Goods Acts may be more applicable, particularly as the body of case law available for the interpretation of such statutes may provide more certainty in particular circumstances. Nevertheless in cases where legislation does not apply, or where novel circumstances arise (as is often the case with the ongoing development of new technology), some reference to the common law may be possible.

\textsuperscript{326} Foley v Classique Coaches Ltd [1934] 2 K.B.
\textsuperscript{327} Aoteaor International Ltd. v Scancarriers A/S. [1985] 1 N.Z.L.R. 513 at 556 (Privy Council)
\textsuperscript{328} Francis v Cockrell (1870) L.R. 5 Q.B. 501
\textsuperscript{329} e.g. Stewart v Ravell’s Garage [1952] 2 Q.B. 545.
This issue is also important in the context of micro-agreements. When a micro-agreement is formed by parties to a large information technology project, additional work may be necessary as a result of the micro-agreement, even though that work was not envisaged under the main contract. The implication of a standard of performance for the additional work may be important to ensure that parts of the system are not introduced with a lesser standard than required contractually under the main contract.

Further discussion on implied terms relating to the standard of performance is provided below in relation to terms implied by statute.

3.4.10  Implying a Term where One Party has already Performed Obligations According to that Term

A Court will be more likely to imply a term into that contract if one of the parties has performed its obligations in accordance with that term. Specifically, Denning J. in Sykes (Wessex) Ltd. v Fine Fare Ltd.\textsuperscript{330} said:

"In a commercial agreement the further the parties have gone with their contract, the more ready are the Courts to imply any reasonable term so as to give effect to their intentions. When so much has been done, the Courts will do their best not to destroy the bargain. When nothing has been done, it is easier to say that there is no agreement between the parties because the essential terms have not been agreed. But when an agreement has been acted upon and the parties, as here, have been put to great expense in implementing it, we ought to imply all reasonable terms so as to avoid any uncertainties."

3.4.11  Duty to co-operate

In an information technology development or systems integration contract, the party contracted to provide the system may also be required to undertake systems analysis work to establish the overall requirements of the system before it is implemented. If the buyer of the system does not co-operate with the developer to provide relevant information about the requirements then the buyer itself may be in breach of an implied duty to co-operate.

In Mona Oil Equipment & Supply Co. Ltd. v Rhodesia Railways Ltd.\textsuperscript{331} stated:

"It is, no doubt, true that every business contract depends for its smooth working on co-operation, but in the ordinary business contract, apart, of course, from express terms, the law can enforce co-operation only in a limited degree - to the extent that it is necessary to make the contract workable. For any higher degree of co-operation the parties must rely on the desire that both of them usually have that the business should get done."

3.4.12  Terms Implied by Statute

3.4.12.1  Trade Practices Act

3.4.12.1.1  Introduction

The Trade Practices Act, 1974 allows the implication of terms into a contract for the sale of goods or services under defined conditions. Specific details of the conditions required for implication, as well as the actual implied terms which may become part of a contract are discussed in a following Section.

\textsuperscript{330}  in Sykes (Wessex) Ltd. v Fine Fare Ltd. [1967] 1 Lloyd's Rep. 53
\textsuperscript{331}  Mona Oil Equipment & Supply Co. Ltd. v Rhodesia Railways Ltd. [1949] 2 All E.R. 1014 per Devlin J.
It should be noted that the *Trade Practices Act, 1974* generally does not preclude any common law rights, including those discussed in previous Sections (s.51AAA)\(^3\). Accordingly, legal rights and obligations under the *Trade Practices Act* generally operate concurrently with other laws.

### 3.4.12.1.2 The Conditions for Implying Terms Under the *Trade practices Act*

Under the *Trade Practices Act, 1974*, terms will be implied into a contract if that contract is for the supply of goods or services by a corporation to a "consumer". The word "Consumer" is a key concept which has a specific meaning under the Act. In summary, a purchaser is a "consumer" if the price of the goods or services purchased is less than $40,000; OR, if the price is more than $40,000 the goods or services were ordinarily acquired for personal, domestic or household use or consumption; AND, the goods or services were not acquired for re-supply or to be used in manufacturing or repair processes. The exact definition of "consumer" under the Act is: \(^3\)

"(1) For the purposes of this Act, unless the contrary intention appears:
(a) a person shall be taken to have acquired particular goods as a consumer if, and only if:
   (i) the price of the goods did not exceed the prescribed amount; or
   (ii) where that price exceeded the prescribed amount the goods were of a kind ordinarily acquired for personal, domestic or household use or consumption or the goods consisted of a commercial road vehicle;
   and the person did not acquire the goods, or hold himself or herself out as acquiring the goods, for the purpose of re-supply or for the purpose of using them up or transforming them, in trade or commerce, in the course of a process of production or manufacture or of repairing or treating other goods or fixtures on land; and
(b) a person shall be taken to have acquired particular services as a consumer if, and only if:
   (i) the price of the services did not exceed the prescribed amount; or
   (ii) where that price exceeded the prescribed amount the services were of a kind ordinarily acquired for personal, domestic or household use or consumption.

(2) For the purposes of sub-section (1):
(a) the prescribed amount is $40,000 or, if a greater amount is prescribed for the purposes of this paragraph, that greater amount" \(^3\)

In relation to this definition the following points are made:

- a "person" includes a Corporation (s.4(5));
- the $40,000 limit appears to relate to the per-unit price rather than the total contract price (although the position is not absolutely clear)\(^3\). Thus a contract for the supply of (for example) one thousand disk drives at $1,000 each would be within the $40,000 limit; and
- the phrase "goods of a kind ordinarily acquired for personal, household or domestic use" is not defined further in the Act. It does not appear however to preclude the business use of such goods. Calvert\(^3\) discusses this and argues that "although the law is not clear, it is likely the term 'ordinarily' means regularly, or not unusually, rather than in the majority of cases. Thus it may be

---

332 *The Trade Practices Act, 1974*, Section 51AAA states: "It is the Parliament's intention that a law of a State or Territory should be able to operate concurrently with this Part unless the law is directly inconsistent with this Part".

333 *Trade Practices Act, 1974*, Section 49 - "Consumers".


335 Calvert, M. *Technology Contracts - A Handbook for Law and Business in Australia*, Butterworths, 1995 at 189-190
argued that personal computers and associated components may come within the definition of "personal, household or domestic" goods.

- if goods and services which fall within the category of "personal, household or domestic use..." then implied terms cannot be contracted out.

The practical effect of the above is that the *Trade Practices Act, 1974* can imply terms into a contract for goods or services. The terms which may be implied are discussed further below.

### 3.4.12.1.3 The Terms Which May be Implied Under the Trade Practices Act 1974

**Goods or Services?**

The *Trade Practices Act, 1974* potentially can imply a more comprehensive set of terms into contracts for goods than contracts for services and these terms are summarised in following paragraphs. In relation to contracts for information technology products however, there is some uncertainty about which category software fits. A detailed analysis of this issue is provided in (e.g.) Akindemowo\(^{336}\) however in summary the following points are made:

- the *Trade Practices Act, 1974*\(^{337}\) provides definitions of both "goods" and "services" although neither of these refer to software.
- some cases have categorised software as services. For example, *Caslec Industries Pty Ltd v Windhover Data Systems Pty Ltd*\(^{338}\) categorised the, the provision of an "off the shelf" software package with incidental services of staff training and enhancement" as a contract for services.
- other cases have categorised software as "goods": e.g. *St. Albans City Council v ICL*\(^{339}\), *Toby Constructions Products Pty Ltd v Computa Bar (Sales) Pty Ltd*\(^{340}\).
- based on an analysis of the cases, commentators\(^{341}\) have suggested that agreements relating to hardware or turnkey computer systems would be considered as contracts for the provision of goods, while off-the-shelf software, or software which is customised, is more likely to be considered as a contract for the provision of services.
- overall, relevant factors to consider include whether any identifiable physical property is to be transferred; whether the overriding purpose of the transaction\(^{342}\) is to procure physical articles (goods) or skill and labour (services); whether the transaction relates to the transfer of software and incidental services only;
- terms can also be implied into contracts by means other than the Trade Practices Act, 1974 (or equivalent). These include common law as well as interpretation of express terms\(^{343}\).

---

\(^{336}\) Akindemowo, Olujoke: *Information Technology Law in Australia*, First Edition, LBC Information Services, 1999 at pages 64-67

\(^{337}\) Section 4. (1)

\(^{338}\) *Caslec Industries Pty Ltd v Windhover Data Systems Pty Ltd* (Federal Court of Australia, Gummow J, No G627/90, 13 Aug 1992, unreported

\(^{339}\) *St. Albans City Council v ICL* (Q.B. Baker J, 3 October 1994, unreported)

\(^{340}\) *Toby Constructions Products Pty Ltd v Computa Bar (Sales) Pty Ltd* (1983) 2 NSWLR 48

\(^{341}\) see Akindemowo, Olujoke: *Information Technology Law in Australia*, First Edition, LBC Information Services, 1999 at page 67

\(^{342}\) this test is also called the "substance of the contract" test and is discussed in Akindemowo, Olujoke: *Information Technology Law in Australia*, First Edition, LBC Information Services, 1999 at page 64

\(^{343}\) In Akindemowo, Olujoke: *Information Technology Law in Australia*, First Edition, LBC Information Services, 1999 at page 67, a number of references were cited for this proposition including: *Saphena Computing Ltd v Allied Collection Agencies Ltd* [1995] FSR 616, and *Beta Computers (Europe) Ltd v Adobe Systems Ltd* 1996
In practice, as the question does not appear to have been definitively settled, care needs to be taken in drafting contracts to ensure that the parties expectations are recorded clearly and that conditions and warranties are specifically identified where appropriate.

In relation to the actual terms which may be implied, if the elements discussed in the preceding Section are met, the Trade Practices Act, 1974 allows for the implication of the following specific conditions and warranties:

**Contracts for Goods**

Implied Conditions:
- Section 69(1)(a) - implies a condition that the seller has a right to sell the goods;
- Section 70(1) - implies a condition that the goods will correspond to the description. (This may be important in the context of information technology contracts where often the "description" of the goods to be provided is contained in a technical specification);
- Section 71(1) - implies a condition that goods are of merchantable quality;
- Section 71(2) - implies a condition that goods are fit for purpose;
- Section 72 - implies a condition that bulk goods will correspond with the sample.

Implied Warranties:
- Section 69(1)(b) - implies a warranty of quiet possession;
- Section 69(1)(c) - implies a warranty that goods are free from encumbrances.

**Contracts for Services**

Implied Warranties:
- Section 74(1) - implies a warranty that due care and skill will be used and that supplied materials are fit for purpose.
- Section 74(2) - implies a warranty that services will be fit for purpose.

**3.4.12.1.4 Technology Contracts and the Trade Practices Act 1974**

The above is intended to provide a summary list of what terms may be available under the Trade Practices Act, 1974. The effect of these implied terms will be the subject of more detailed discussion in the context of specific examples provided in the following chapter "Micro-Agreements Part 1: Concepts and Case Studies".

It seems reasonably clear that under some circumstances the Trade Practices Act, 1974 will imply conditions or warranties into information technology contracts. For example, the supply of contract programming services up to an amount of $40,000 may result in terms being implied through the operation of that Act.

In larger software developments and systems integration contracts however it is likely to be more difficult to imply such terms because firstly, the cost of individual components is more likely to exceed the prescribed amount of $40,000; and secondly, a "hierarchy" of contractors and sub-contractors supplying components to a systems integrator is likely to do so in the context of "re-supply" or "manufacture", and accordingly the implication of conditions and warranties under the Trade Practices Act, 1974 will be prevented.

**3.4.12.2 Sale of Goods Act**

All Australian states and territories have Sales of Goods legislation which allows terms to be implied into contracts for the sale of goods, although this is subject to the contrary intention of the parties.
As an example, the *Sales of Goods Act 1923* (NSW) allows the following terms to be implied:

**Implied Conditions:**
- Section 17(1) - implies a condition that the seller has a right to sell the goods;
- Section 18 - implies a condition that the goods correspond with the description;
- Section 19(1) - implies a condition that 'where a buyer makes known to the seller ... the particular purpose for which the goods are required so as to show that the buyer relies on the seller's skill or judgement, and the goods are of a description which it is in the course of the seller's business to supply (whether the seller be the manufacturer or not), there is an implied condition that the goods shall be reasonably fit for such purpose';
- Section 19(2) - implies a condition that 'where goods are bought by description from a seller who deals in goods of that description (whether the seller be the manufacturer or not), there is an implied condition that the goods shall be of merchantable quality';
- Section 20(2)(a) - implies a condition that, where goods are supplied by sample, the bulk will correspond with the sample.

**Implied Warranties:**
- Section 17(2) - implies a warranty that the buyer shall have and enjoy quiet possession of the goods.

### 3.4.12.3 Other Legislation

Apart from the *Trade Practices Act, 1974* and the *Sales of Goods Act* mentioned above, other legislation such as the *Conveyancing Act 1919* (NSW) also provides for the implication of terms into contracts\(^{345}\).

### 3.5 UNJUST ENRICHMENT, QUANTUM MERUIT, MONEY HAD AND RECEIVED

#### 3.5.1 Quantum Meruit

A claim in quantum meruit is a restitutionary remedy in its own right\(^{346}\). It involves "the payment of an amount which constitutes, in all relevant circumstances, fair and just compensation for the benefit or 'enrichment' actually or constructively accepted"\(^{347}\). Ordinarily, that will correspond to the fair value of the benefit provided (for example, remuneration calculated at a reasonable rate for work actually done or the fair market value of materials supplied).\(^{348}\)

The basis of the claim is founded in the doctrine of unjust enrichment and as such "the right to recover does not depend upon the existence of an implied contract but on a claim to restitution based upon unjust enrichment"\(^{348}\). Thus it is not necessary to show that a contract existed, but rather that one party was unjustly enriched at the expense of the other, provided the work done was not of the kind that is normally undertaken gratuitously.

---

\(^{345}\) Including general definitions such as "person" (which includes a corporation), and "month" which means a calendar month.


\(^{347}\) See *Craven-Ellis v Canons Ltd* (1936) 2 KB 403 - another quantum meruit case expressly based upon unjust enrichment.


---

Paul P. Parnell 79
In summary, a claim may be available "where one party does work for the benefit of itself or another on the assumption that a contract would ultimately be made between them and where the second party unilaterally abandons the project"\(^3\)\(^5\)\(^0\).

Despite the above Deane J. placed limits on the amounts claimable when he stated that the remuneration must "satisfy the requirements of good conscious and justice which inspire the concept or principle of unjust enrichment"\(^3\)\(^5\)\(^1\). Note however that the amount claimed under quantum meruit may in fact be greater than any contracted price\(^3\)\(^5\)\(^2\).

Overall, in large information technology project systems integration projects it is unlikely that all work undertaken would be explicitly covered by a contract and accordingly, to be fair to all parties, care needs to be taken to ensure that all parties are aware of their respective rights and obligations in relation to such work.

### 3.5.2 Money Had and Received

The action of money had and received is available when one person pays money to another under a mistake of fact, when that money would not have been paid if the mistake had been known at the time\(^3\)\(^5\)\(^3\). This action is based on the doctrine of unjust enrichment\(^3\)\(^5\)\(^4\) and in a joint judgement the High Court held that:

"Before that prima facie liability [to make restitution] will be displaced, there must be circumstances (for example, that the payment was made for good consideration such as the discharge of an existing debt or, arguably, that there has been some adverse change of position by the recipient in good faith and in reliance on the payment) which the law recognises would make an order for restitution unjust"\(^3\)\(^5\)\(^5\).

### 3.6 CONCLUSION

In conclusion, this chapter has attempted to give an overview of a number of issues associated with dealings in the context of large information technology projects.

In summary, parties to commercial transactions involving the development or acquisition of these large information technology based systems need to be cognisant of all the legal factors which may affect their rights and obligations. Some of these are more obvious and self contained, such as terms expressed in a written contract for the supply of a computer systems, while others are not so obvious to the uninitiated. Issues of estoppel, unconscionability, and implied contractual terms, to name a few, may produce unintended consequences for the unwary. As will be seen in the following chapter, seemingly "innocent" arrangements between the parties involved in information technology contracts may lead to significant problems. From the discussion presented in this chapter it is seen that the legal system in its present form may bring a variety of concepts and devices to the table which may affect the result of commercial dealings.

\(^3\)\(^5\)\(^0\) Sabeno Pty Ltd v North Sydney Municipal Council [1977] 2 NSWLR 880; British Steel Corporation v Cleveland Bridge and Engineering Company Ltd [1984] 1 All ER 504
\(^3\)\(^5\)\(^1\) Pevey and Mathews Pty Ltd v Paul (1987) 162 CLR 221
\(^3\)\(^5\)\(^2\) Renard Constructions (ME) Pty Ltd v Minister for Public Works (1992) 26 NSWLR 234.
\(^3\)\(^5\)\(^4\) Australia and New Zealand Banking Group Ltd. v Westpac Banking Corp. (1988) 62 ALJR 292, per Mason C.J., Wilson, Deane, Tooley and Gaudron JJ.
\(^3\)\(^5\)\(^5\) Australia and New Zealand Banking Group Ltd. v Westpac Banking Corp. (1988) 62 ALJR 292, per Mason C.J., Wilson, Deane, Tooley and Gaudron JJ.
Overall, the law strives to be fair and equitable in its approach however despite this it is constantly evolving to meet the needs of commercial enterprises, and in doing so can be complex in its detail. The practical application of these and other issues is provided in the following chapter where specific case studies are presented.
The greatest amount of care has been taken while scanning the following pages. The best possible results have been obtained.
4. “Micro-Agreements”
Part 1: Concepts and Case Studies

The objective of this chapter is:
- to introduce the new legal concept of “micro-agreements” and identify the benefits and disadvantages of micro-agreements through a number of case studies specific to the information technology industry
- to provide recommendations for maximising the benefits of micro-agreements and minimising their disadvantages

4.1 INTRODUCTION
In larger information technology systems integration projects, many individuals and organisations may be involved in the engineering and other activities associated with the development. Extensive day to day communication between the parties is usually vital to ensuring that the functionality, performance and interfaces of each of the software and hardware components match up. As a consequence, it is likely that many “micro-agreements” are made between individuals and teams both formally and informally over the life of the project.

As an overall aim, this chapter examines these “micro-agreements”, (and the conduct of the parties generally), to determine the effect on legal rights and obligations. A number of case studies are provided and particular problems associated with each micro-agreement are examined. An analysis of each case study is made with the aim of identifying solutions based on the law as it currently stands. This is done by firstly examining how the contracting process itself can be improved - the primary focus being how to link the contractual arrangements to current software engineering “best practice”; and secondly, by highlighting potential inadequacies in the law itself. Overall, suggested solutions to the problems identified through the case studies are also provided.

Overall, this chapter looks in detail at micro-agreements in the context of some of the fundamental “best practise” areas within the software engineering field. This approach allows relevant problems to be identified and analysed to the extent necessary to extract trends and make recommendations on how to maximise the benefits and minimise the adverse affects of micro-agreements.

The next chapter, Relationship Models, puts micro-agreements into a broader context by identifying how to best structure a legal relationship between the various legal entities to achieve the optimum result from the software integration perspective.
4.2 METHOD OF ANALYSIS AND EVALUATION

In order to provide a thorough evaluation of micro-agreements, a systematic approach is required. The process of undertaking an orderly analysis of the law has been summarised (in the context of judge made law) as follows:

"... Thus the joint judgement, by a process of induction, arrives at a general statement of principle consistent with the authorities discussed but one capable, because of its greater generality, of application to a new class of facts. The mere capability of application to a new class of facts is, however, not enough. Two further steps are required. In the first place, the fundamental policy which drives the Court to interfere in situations it previously avoided must be discovered and applied to the principle. Second, the objections which previously held back the proposed extension of principle must be answered. These steps are required if the Court is to conform to the accepted methods of orderly development of legal principle."

Although this statement applies to judge made law, when the essential elements are distilled the following basic steps may be used to undertake any legal analysis:

- identify a general statement of principle of the law in the particular areas which are applicable to the class of facts under consideration
- establish whether each principle is in accord with fundamental policies, and
- answer objections to the application of the principle

Using the above points as a guide, the following method of approach was used in the analysis of micro-agreements:

(a) Provide a definition of "micro-agreement" and give a general statement of principle
(b) Present examples and case studies of micro-agreements which are grouped logically in the context of various software development and acquisition activities. For each case study:
   - identify the legal principles applicable to the particular case study;
   - identify the potential problems which may arise as a result of the micro-agreement; and
   - Provide recommendations on how to prevent each problem from occurring.

The ultimate aim of the steps referred to above is to provide a relatively comprehensive assessment of "micro-agreements" in the information technology systems integration context. It also aims to provide practical recommendations on how to avoid a range of problems associated with the Systems Integration contracting process.

4.3 DEFINITIONS

4.3.1 Micro-agreement

As the term "micro-agreement" is original to this work and is intended to describe a new legal concept, the following definition is provided.

"micro-agreement" means:

"an agreement or common understanding between two or more parties involved in an information technology or related development or acquisition project, which relates to some aspect of work associated with that project, but is not necessarily expressly or by implication covered by any existing or planned contractual clauses".

The micro-agreement may not necessarily be supported by consideration and accordingly may not necessarily constitute a contract in itself.

Large complex systems generally need high levels of communication between the parties involved so that mutual aims can be clearly identified and issues resolved. Apart from the legal definition provided above, in a practical sense the term is meant to provide a general way of describing much of the day to day discussions, documents, meetings and interactions, both formal and informal, of individuals and organisations involved in developing complex systems.

As an example, two software developers from different companies may together agree on certain interface attributes of two software components being developed which need to be integrated together. These new interface attributes may not strictly conform to a published interface specification, or one contained in a formal contract specifying the work. The micro-agreement in this case constitutes a verbal agreement between two individuals which may be necessary or desirable to achieve an optimum technical solution, but which is not strictly covered by the prime contract or specification.

Detailed examples of a number of different types of micro-agreement are provided below in the context of activities often associated with software development and systems integration work. The examples identified are intended to provide coverage important activities associated with the software development process. To that end, at least one example of a micro-agreement is provided for each of a number of software engineering process elements. Appendix C provides a full list of activities which are recognised as being conducive to good software engineering practice.

4.3.2 Systems Integration

"Systems Integration" has been defined as:

"The service which draws together the various elements required to provide a comprehensive information processing solution, through a combination of professional service and expertise". 357

A systems integrator is thus the entity responsible for connecting together a number of sub-components into a fully functioning unit capable of satisfying customer requirements. The systems integrator may or may not be the ultimate end user of the system and thus may or may not be the subject of a contract between itself and the buyer.

4.3.3 Software Engineering

"Software Engineering" is defined as:

"A logical process of activities which transforms a set of requirements arising from a specific mission objective into a full description of a system which fulfils the objective in an optimum way. It ensures that all aspects of a project have been considered and integrated into a consistent whole." 358

357 See Section 2.8 above.
358 See Section 2.3.2 above.
4.4 LINKING THE SYSTEMS INTEGRATION CONTRACT TO SOFTWARE ENGINEERING “BEST PRACTICE”

4.4.1 Introduction

In the Sections which follow, a number of case studies and examples are provided where a micro-agreement has been formed which has had a significant effect on the outcome of the particular project. In order to show how problems associated with the micro-agreement may have been avoided in the first place, an analysis of the facts and the law is conducted and a number of specific recommendations made. As an illustration of this, quite often it is shown in the following case studies that the adverse affects of micro-agreements may have been avoided if software engineering “best practises” were followed. One of the main recommendations therefore is that existing laws be used to enforce such software engineering “best practices” where appropriate. It is considered by the author that the information technology contracting process has not always taken full advantage of such “best practice”. An effective software engineering process can itself provide a powerful tool for resolving issues, uncertainties and disputes and if such a process is contractually binding to some degree then it may provide a better result overall (although of course cost and schedule implications must also be addressed in context).

In relation to software engineering process generally, the software engineering community has, over the years recognised that software is not an exact science and as such needs special attention in the processes which are used during development. These processes, although not necessarily ideal for every situation, are at least reasonably well documented\(^\text{359}\) and continue to evolve as better methods, tools and practices are identified.

![Diagram: Link the contract to Software Engineering “best practice”](image)

**Figure 12: Link the contract to Software Engineering “best practice”**

4.4.2 What is Software Engineering “Best Practice”?

There is no generally recognised scientific definition of “best practice” however for the purposes of this document it is assumed to be a method of conducting an activity in such a way as to provides positive benefits when compared to other practices. A “best practice” for the purpose of this document is a practice which provides *measurable* benefits and for which there is quantitative research data to show this.

Software Engineering “best practice” will vary depending on many factors including the type of development underway, the size of the project and the technologies involved. By

identifying individual "best practices" therefore there is risk that they may not be applicable to each individual development, and in addition, each "best practice" may be superseded over time as new developments, technologies and processes come into existence or evolve.

Due to the nature of software and the problems associated with such developments (some of which were outlined in Chapter 2, *The Nature of Large Information Technology Projects*), the software community has itself attempted to understand the reasons why such problems occur and to identify solutions.\(^{360}\) One example of a set of recommended "best practices" is provided by McConnell\(^ {361}\) and itemised briefly below. Brief descriptions of each of these is provided in Appendix D - Software Engineering Best Practices". McConnell recommends the following processes, techniques etc. for achieving optimum results:

- Change control boards,
- Daily Build and Smoke Tests,
- Designing for Change,
- Evolutionary Delivery,
- Evolutionary Prototyping,
- Goal Setting,
- Inspections,
- Joint Application Development (JAD),
- Lifecycle Model Selection,
- Measurement,
- Miniature Milestones,
- Outsourcing,
- Principled Negotiation,
- Productivity Environments,
- Rapid-Development Languages,
- Requirements Scrubbing,
- Reuse,
- Signing Up,
- Spiral Lifecycle Model,
- Staged Delivery,
- Theory-W Management,
- Throwaway Prototyping,
- "Timebox" Development,
- Tools Group,
- Top-10 Risks List,
- User-Interface Prototyping,
- Voluntary Overtime.

Although the above list identifies important "tools" which can be used to maximise the chance of a successful outcome, they are intended to focus on the *rapid* development of systems and to some degree go beyond the more fundamental or basic practices which will lead to general improvements in schedule, cost and quality. Note that it is not intention at this point to give a blanket recommendation (or otherwise) on the use of any or all of these particular techniques to solve systems integration problems in a general sense.

In contrast to the above list of "best practices", research has also identified "worst practices" including\(^ {362}\):

---


---

Paul P. Parnell

86
- Having no historical software-measurement data
- Rejecting accurate estimates
- Failing to use automated estimating tools and automated planning tools
- Allowing excessive, irrational schedule pressure and creep in user requirements
- Failing to monitor progress and to perform formal risk management
- Failing to use design reviews and code inspections

In a simplistic sense, if an information technology contract took each of the above issues into account and imposed explicit legal rights and obligations in accordance with these practices then theoretically there would be a good chance of achieving the desired outcome.

For the purposes of this work however, rather than focusing individually on the many best practices and worst practices listed above, the focus is on a minimum set of six fundamental practices. Each of these fundamental practices is discussed in detail in a separate case study in this chapter.

The set of fundamental practices have been derived from the Capability Maturity Model. Figure 1: Software Engineering Capability showed the overall framework for an effective software engineering capability and this framework is consistent with the Capability Maturity Model.

This model was designed to allow the assessment of an organisation’s software engineering capability by identifying the presence or absence of specific desirable practices. This model effectively identifies “best practices” in the software engineering discipline (although they are not actually called “best practices, but are rather grouped as “Capability Maturity Levels”). Briefly, the five levels of “capability maturity” are identified as: Level 1 - Initial, Level 2 - Repeatable, Level 3 - Defined, Level 4 - Managed, Level 5 - Optimised.

The figure below shows that the “capability” or effectiveness of an organisation’s software engineering work can be ‘measured’ or in fact defined in accordance with organisational goals in relation to cost, schedule and quality.

Figure 13: Software Engineering Capability Maturity Levels

For each of the Software Engineering Capability Levels shown in the above diagram, the Capability Maturity Model identifies fundamental or "key" practices in such a structure as shown below:

- Organisational Structure
- Resources, Personnel and Training
- Technology Management

- Documented Standards and Procedures
- Process Metrics
- Data Management and Analysis
- Process Control

These issues are addressed for each Capability Maturity Level

Figure 14: Capability Maturity Levels - Key Practices

The Capability Maturity Model: "Level 2 - Repeatable" level of the Capability Maturity Model provides an effective set of "best practices" which are important to the successful development of any significant software project. In this chapter the focus is on these Level 2 practices which, if it is submitted, should be built contractually into a development or systems integration activity where possible, to provide an effective software engineering

---

process capable of handling many of the day to day difficulties which occur. Each of the case studies in this chapter are intended to focus on one major process or fundamental practice. In summary, the fundamental practices consist of:

- Requirements management
- Software project planning
- Software project tracking and oversight
- Subcontractor management
- Software quality assurance
- Software configuration management

This set constitutes a sub-set of "key practices" identified by the Software Engineering Institute\(^{365}\) in its development of its software "Capability Maturity Model". The stated objective of the Software Engineering Institute was to "provide a standard [software capability maturity] model that is based on actual practices and that is documented and publicly available"\(^{366}\). The work involved collecting data on many software projects using questionnaires, surveys and advice from practitioners generally in an attempt to distil the essence of good software engineering practice into a manageable and practical form.

The above six practices are intended to represent the basic set of processes needed by an organisation to achieve a level of "repeatability" across software development projects\(^{367}\). In other words, unless these practices are performed, there is likely to be less consistency as well as higher risk in terms of cost, quality and schedule. Of course some individual projects may be very successful without all of these practices in place but overall the development an integration activities should be less risky if they are performed. In addition, as each of the above practices has a cost associated with it, individual projects need to weigh up the cost/benefit of enforcing such practices.

What evidence is there that these fundamental practices improve the software development process? The answer to this question was summed up by the Software Engineering Institute as:

"There are dozens of sources of theory and practice that describe the benefits of improving process capability... Case studies from the software engineering community and elsewhere suggest that addressing issues of process management, measurement, and institutionalization improve the organisation's ability to meet its cost, quality and schedule goals"\(^{368}\).

Specific improvements have been documented as\(^{369}\):

- Defect reduction of up to 90%
- Productivity gains of up to 350%
- Schedule reduction of up to 70%

\(^{365}\) Carnegie Mellon University

\(^{366}\) Software Engineering Institute: A Systems Engineering Capability Maturity Model, Version 1.1, SECMM-95-01;CMU/SEI-95-MM-003, November 1995 at page 0-8

\(^{367}\) Software Engineering Institute: A Systems Engineering Capability Maturity Model, Version 1.1, SECMM-95-01;CMU/SEI-95-MM-003, November 1995 at pages 0-14, 0-15


\(^{369}\) see Section 2.3.2 above.

Paul P. Parnell 89
Despite the above, it should be recognised that the approach of linking the contract to software engineering best practice is no panacea and is based on a number of assumptions.

Overall it is submitted that in order to effectively deal with problems which arise in systems integration contracts, not only should the contract provide the more traditional “Dispute Resolution” clauses for when things start to go wrong, but also should contractually enforce a solid software engineering process which itself is capable of dealing with such problems, ideally on a basis of providing continuous monitoring of progress and early warning of potential problems.

4.5 EXAMPLES AND CASE STUDIES - SYSTEMS INTEGRATION

This Section focuses on micro-agreements in the context of software development and systems integration activities.

The case studies are intended to provide examples of where a micro-agreement has potentially a significant impact on the overall progress and success of software development and systems integration activities for a particular project. As discussed above, the case studies illustrated below are grouped according to activities associated with a recognised software development process model - the “Capability Maturity Model” which has a degree of visibility and prominence in the software community. The main reason for using this grouping is to provide a reasonable coverage of the software development process and provide at least one example of a micro-agreement within each type of activity associated with the development process. The Case Study headings are designed to cover the activities generally recognised as being most important or fundamental to the success of software development projects. It should be recognised that other activities are also undertaken in the software engineering world and although they have not been explicitly covered by the case studies, some of these additional practices are summarised in Appendix C - List of Software Engineering Activities.

All following case studies are fictitious but are based on possible real-life scenarios. Some case studies are simplified to the extent necessary to focus on the particular software development or integration activity in question. In practice, many of the case studies would be complicated by factors external to the particular activity in question. The legal analysis of each case study is not intended to be exhaustive, but rather to give sufficient coverage of the issues in question.

---


CASE STUDY 1 - MICRO-AGREEMENTS IN THE CONTEXT OF REQUIREMENTS MANAGEMENT

4.5.1 Introduction and Explanation of the Activity

The process of building a software based system generally involves identifying the requirements of that system. Simplistically, the requirements are aimed at providing a statement of what the system is required to do. Requirements generally include such things as the functions which the software has to perform, the performance of the system, and a definition of the interfaces between the different components. The requirements can also include "business" needs such as cost, schedule and quality requirements.

In a large system, the number and scope of requirements can be extensive and continually changing (Section 2.5 Requirements Management provided a discussion on the extent to which requirements can change over the life of a project). The process of managing requirements has been stated as follows:

"Requirements Management involves establishing and maintaining an understanding and agreement with the customer on the requirements for the software throughout the software lifecycle. The agreements cover both the technical requirements for the software and the non-technical requirements, such as delivery dates for the software. The agreements form the basis for estimating, planning, performing and tracking the project's software activities."372

4.5.2 Example of a Micro-agreement

The following is a specific scenario which may occur in the context of a large systems integration contract, and which focuses on requirements management issues.

The development of a large commercial communications system is currently underway. The system uses the latest technology and software design techniques and involves the interconnection of six major software components via a wide area network using the ATM373 communications protocol. Each of the major components runs on a separate computer and ultimately will reside in a different physical location. The buyer has taken on the risk of integrating these components into a single working system.

---

373 ATM: Asynchronous Transfer Mode is a communication protocol allowing the transfer of digital data in small packets. The protocol is designed to allow the integration and transmission of different types of data such as voice, video, and data as an integrated high capacity transmission.
The buyer, as Systems Integrator, has let contracts to each of six companies for the development of the components of the system. Each contract includes a detailed specification which explicitly defines the requirements which include (inter alia) the interface data formats between each of the components.

The interface definition between each component was provided in the form of a specification written in the CORBA IDL\textsuperscript{374} software language and accordingly, the interface definition attempted to provide a rigorous, complete and compilable\textsuperscript{375} specification.

During the developmental stages of the project, two employees who worked for Company A and B respectively, had the task of integrating software components together using the standard interface definition provided in the contract. As skilled and experienced software developers however they together discovered that a minor change to the interface specification would be likely to improve the overall system performance and simplify the integration task. A micro-agreement was thus formed between these two employees that each would work to the improved interface. As the interface would be eventually compiled into the system and kept under formal configuration control, and to some extent as it was still in the developmental stages and hence subject to possible further changes, neither thought it was necessary or prudent to formalise the change until after the integration work became reasonably stable. Both understood that to change the prime specification at this stage would take significant time, effort and administrative overhead and was also likely to lead to a formal contract change with subsequent possible claims for cost and schedule extensions. During discussions between the two employees however the employee of Company B said to the employee of Company A:

"I have made some minor changes to the interface which will improve its performance and ease of implementation. If you agree to use this modified

\textsuperscript{374} CORBA - Common Object Request Brokered Architecture; IDL - Interface Design Language. CORBA IDL is a human readable computer language which can be used to describe the interfaces between computer systems. It is intended to provide a rigorous definition of interface components and be independent of specific computer hardware attributes.

\textsuperscript{375} The process of converting computer software "source code", which is human-readable, into a machine readable form is called "compilation".
interface then I'm sure you will see similar benefits. Once we prototype the changes and confirm that they work I will later arrange to update the interface specification formally".

Due to the size of the project and the physical separation of the different companies involved, it was overlooked by the two employees that a third software component was being built by Company C which also needed to interface with their components. That third company produced its software component in accordance with the original interface specification defined in its contract with the Systems Integrator, however it was found during integration testing that communication between components was not possible because of the minor interface changes.

The interface test failed and the component from Company C was initially rejected. The Systems Integrator refused to give Company C a milestone payment because of the test failure. Company C undertook additional work to verify and demonstrate that its interface was in accordance with the contract. Despite the fact that the new interface was a technical better solution, in order to complete the initial contract, both Company A and B changed the interface back to that which was identified in the original contract. This resulted in additional work for all parties and major parts of the formal integration testing needed to be repeated.

The micro-agreement in this case study thus consisted of an agreement between an employee of Company A and an employee of Company B to implement a slightly modified interface rather than the one specified in the prime contract.

4.5.3 Potential Problems arising from the Micro-agreement

The above is a simple example of where, in a complex development, a micro-agreement between two individuals can have a significant impact on the final product to be delivered. Issues which are important in the context of this chapter include:

1. Despite that fact that the interface specification was rigorously defined in the contract, for technical reasons it was desirable for minor changes to be made to it after the contract was in place.

2. The end result of the micro-agreement was that three of the subcontractors needed to perform additional work (with associated cost and schedule increases). This was despite the fact that if the micro-agreement was put into effect (i.e. with the interface modifications), the final product would have been technically superior and the integration activity simpler for all parties.

3. As a consequence of the micro-agreement Company C performed additional work to verify that its component conformed to the contracted interface specification. The test specification used to verify compliance with the contracted specification identified that components A, B, and C did not communicate correctly, however it did not identify which component was at fault.

4. After re-work of the interface, formal integration testing needed to be repeated. This required additional work and cost for all parties. The System Integrator applied pressure to Company C to re-run integration tests which had failed because of the changes made by Company A and B.
5. Due to the additional re-work and testing of the interfaces, the product was delivered later than expected.

4.5.4 Avoiding the Problem

In the context of "Requirements Management", it seems clear from this example that there was not a sufficiently adequate mechanism in place to allow changes to requirements (in this case interface requirements) to be easily made to the contract. Although it is likely (and relatively common) for a software development or systems integration contracts to provide a formal method for contracting changed requirements to the system, the practical day to day reality of the situation is not always adequately addressed.

The traditional way of allowing changes to be incorporated into a contract is to provide a clause to the following effect:

"The Systems Integrator or the Developer may propose changes to the Specification at any time. Both parties must agree in writing to the changes before commencement of any work associated with such changes".

Another example of this type of clause can be found in Calvert:376

"1. CHANGES TO THE SPECIFICATION
1.1 Customer may request changes (including deletions and additions to the Specification) by notifying Developer of:
(a) the proposed changes to the specification; or
(b) a detailed explanation of the proposed changes to the Technology.
1.2 Developer must promptly:
(a) review each request submitted under clause 1.1;
(b) notify Customer whether Developer is willing to make the changes; and
(c) if Developer is willing to make the changes, prepare and submit to Customer, at Developer’s cost, a written submission including:
(i) the proposed changes to the Specification;
(ii) any required changes to the Project Plan; and
(iii) any required changes to the Budget...."

Sometimes additional words to the following effect are also provided in the contract:

"The Systems Integrator shall not require that the Developer perform any work which is not explicitly identified in the Specification unless the additional work is agreed to in writing by the Developer. If the Developer performs additional work to that explicitly identified in the contract, the Developer will not be paid any additional amount for such work unless agreed to in writing by the Systems Integrator".

In the context of the current example, if the above clauses were incorporated into the contract and adhered to then (in theory) the problem would not have occurred. This is because the engineers who identified the requirements change would have passed the information to the "contracts department" for incorporation into the contract for subsequent dissemination to all parties. In a larger project it is likely that contracts will be drafted and reviewed by legally trained personnel and would include clauses of the type referred to above. In practice however, in a large project, the individual software developers may be to some extent "isolated" from the contract by one or more layers of management as well as their concentrated focus on trying to get a complex system to work. It is likely that they understood that any changes to the formal requirements specification needed to be formally identified, written, reviewed, passed through management, approved by the contract department etc. etc. As many changes are likely during the development of a

system, they felt it better to complete the work effectively as a prototype to ensure technical adequacy then "retrofit" the changes back into the contract at a later stage.

It general therefore clauses such as the above, while probably necessary, may not be entirely adequate on their own to solve the problem, and it is submitted that in many instances, more is needed.

To minimise the problems and maximise the benefits of this particular micro-agreement an effective requirements management process should have been in place because the requirements management function in the above example was obviously not adequate. One way of providing support for this would be to specify that all work conforms to appropriate developmental standards. Appendix A provides some examples of software related standards which are readily available in the industry through standards organisations. Naturally each software development or systems integration project will need to assess the appropriate standards individually however as an example, ISO/IEC 12207 Information Technology - Software Life-cycle Processes, 1995-08-01 requires that:

"5.3 The developer shall participate in system requirements analysis in accordance with the following requirements:

   Note: If a system is developed in multiple builds, its requirements may not be fully defined until the final build. The developer's planning should identify the subset of system requirements to be defined in each build and the subset to be implemented in each build. System requirements analysis for a given build should be interpreted to mean defining the system requirements so identified for that build.

5.3.1 Analysis of User Input. The developer shall participate in analysing user input provided by the acquirer to gain an understanding of user needs. This input may take the form of need statements, surveys, problem/change reports, feedback on prototypes, interviews, or other user input or feedback.

5.3.2 Operational Concept. The developer shall participate in defining and recording the operational concept for the system...

5.3.3 System Requirements. The developer shall participate in defining and recording the requirements to be met by the system and the methods to be used to ensure that each requirement has been met...

   Note: If a system consists of subsystems, the activity in 5.3.3 is intended to be performed iteratively with the activities in 5.4 (System Design) to define system requirements, design the system and identify its subsystems, define the requirements for those subsystems, design the subsystems and identify the components, and so on".

The above is only an extract of one part of the standard and other related information is also provided in that document. The effect of such clauses is often that the developer is provided with a practical checklist of activities aimed at ensuring requirements are effectively managed. The concept of evolving requirements is also taken into account by the identification of "multiple builds" and the provision of support for iterative development schemes.

The above clauses provide one way of enabling the Requirements Management function to be undertaken with some iterative or evolutionary development. Regardless of which

---

377 Note that paragraph numbering in the following paragraphs is taken from the Military version of ISO/IEC 12207, namely MIL-STD-498, Software Development and Documentation, 5 December 1994.
standards are applied however, it is recommended that the contract should ensure that an effective Requirements Management function is undertaken, either explicitly, or by reference to such standards. In summary the contract should ensure that at least the following activities are covered\textsuperscript{378}:

- The requirements should be documented in a consistent format and should be clearly stated, verifiable and testable.
- A Software Engineering Group should review and agree to the requirements before they are incorporated into the software efforts.
- The requirements should form the basis for the software plans, products and activities.
- Changes to requirements should be appropriately reviewed and incorporated into the software efforts.

The importance of explicitly referencing appropriate standards in the contract is that process issues are less likely to be bypassed or shortcut due to schedule or cost pressures. In systems integration work in particular, the nature of software development is such that it is often difficult to estimate the total amount of work involved. As a consequence the systems integrator is likely to bear the brunt of schedule slippages and cost overruns which may already exist in sub-contracted components of the system. By incorporating the software process into the contract, additional weight is provided to these activities and they thus become more than merely an \textit{internal} developer activity (which internally can be minimised or eliminated if schedule or cost pressures dictate).

Chapter 2 \textit{The Nature of Large Information Technology Projects} also provided a discussion on the identification of Requirements and identified the main problems associated with such activities. A number of specific approaches on how to avoid or minimise the problems were made (see \textit{How to Minimise Changes in Requirements}. Section: 2.5.5) and these included such activities as prototyping, joint application development, and requirements inspections. It is recommended when drafting contractual clauses or identifying standards for software process control that the items listed in that Section be evaluated for possible use. The contract and software engineering process should provide support for such activities where possible. In the context of this case study, a contractual mechanism for allowing the development of a prototype system may have been particularly useful to identify and incorporate required interface requirements.

An example of an attempt to provide support in the contract for a software development process covering change management procedures can be found in the \textit{Government Information Technology Contract} (GITC) clauses provided below.

\textit{“Methodology for Software Development Procedures and Documentation}

120.1 The Contractor shall establish procedures, complying with the methodology (if any) specified in the Official Order:

(a) to identify and control software components of, and changes to, the Developed Software and to maintain the integrity and traceability of the Developed Software at all stages of the development; and

(b) to control the content of the Documentation relating to the Developed Software that is to be delivered under the Contract.

120.2 The procedures referred to in subclause 120.1 must include methods for:

(a) identifying and locating revisions of the software as it is developed;

(b) ensuring compliance with the change control requirement as set out in
subclauses 75.3 to 75.5 \(^{379}\),
(c) controlling the issue of development revisions of the software; and
(d) identifying the extent of the performance towards the Project, Implementation
and Payment Plan. \(^{380}\)

Although the above clause provides for the management of changes to some aspects of
the software, it is submitted that they primarily apply to the management of software
products such as the source code rather than the requirements (although there is a
passing reference to "documentation" which could include requirements documentation). It
is suggested that further refinement may be needed to cover all the points identified above
and in particular, explicit reference to procedures for managing and controlling changes to
requirements may be useful.

Despite the above however it should be recognised that standards and processes are
generally written in a way which attempts to provide the best solution in a general sense
and thus they may not be entirely suitable in individual cases. In addition, standards may
be expensive to implement and care is needed to ensure that they are tailored
appropriately to the particular circumstances. If not, it is possible to end up with a very
expensive development which produces a "wall" of documentation but an unsatisfactory
product. It is submitted that when specifying standards it is important not to "blindly" call
up a particular software engineering standards in a contract without knowing in detail the
full implications of doing so.

In summary therefore it is submitted that the problems described in this case study may
have been avoided or minimised if an effective requirements management function was in
place. Due to the nature of requirements identification and the potential problems which
were described in Chapter 2, consideration should be given to enforcing this function
contractually. This is important in any information technology contract and particularly in
larger systems integration contracts.

4.5.5 Legal Analysis of Problems

Although the above discussion is aimed at identifying possible ways to prevent the
problems from occurring in the first place, if the circumstances described in this case
study did arise, what legal solutions would be available?

4.5.5.1 Summary of Legal Claims

Based on the law as it currently stands, a number of claims and counter-claims are
possible by each of the parties involved in the development. These are briefly summarised
(in point form) below with a more detailed discussion provided on each relevant area of
law in the Sections which follow.

4.5.5.1.1 Possible Claims by the Systems Integrator

From the point of view of the Systems Integrator, it eventually received what was
contracted although it was delivered later than originally expected. The loss suffered by
the Systems Integrator included:

- a loss of profits from its inability to deliver the fully integrated system on time to
  it's customer; and
- a lessening of its reputation in the industry for delivering high quality systems
  on-time and on-budget.

\(^{379}\) these clauses are similar to the ones listed above which provide the basic mechanism for enabling
changes to be incorporated into the contract.

\(^{380}\) URL: http://www.gems.gov.au

Paul P. Parnell
When the Systems Integrator discovered that the interface was modified it put pressure on Company A and B to rectify it to conform to the original specification. The Systems Integrator did not attempt to terminate the contract at that stage but rather, is likely to have issued a “notice to complete” or other communication requiring performance of the contract by Companies A and B.

Possible claims therefore by the Systems Integrator may include:

- A claim that Companies A and B misrepresented to the Systems Integrator that they had the necessary professional software engineering expertise to develop the system and manage the requirements effectively.
- A claim that Companies A and B were in breach of an implied contractual term that an effective requirements management process would be in place within each company.
- A claim that Companies A and B were professionally negligent in that an effective requirements management process was not put in place within each company.

4.5.5.1.2 Possible Claims by Companies A & B

In relation to the two parties who modified the interface in the first instance (Company A and B), as a result of the modification the components did not integrate successfully with the component from Company C, and thus the interface needed to be re-worked to enable it to conform with the originally specified interface. To some extent this rework may have been a direct result of the inability to effectively manage the requirements appropriately.

Overall, it is unlikely that a claim could be made by Company B based on the facts of this case study. Both Company A and B had contracted separately with the Systems Integrator to provide components which conformed to a clearly specified interface. Unless there are some additional facts which could show (for example) they were misled by a pre-contractual misrepresentation, it is likely that despite an honest attempt to improve the product, it is submitted that they would have no claim for the additional work conducted.

The employee of Company B made the statement that once the prototype was proven, he would arrange to have the specification formally updated. Company A may claim that this statement was the basis of its decision to incorporate the changes to the interface and as such may claim that:

- Company B misrepresented that it would formally arrange to update the specification but did not do so;
- Company B should be estopped from denying that it intended to update the specification; and
- Company B, by its false statement (to the effect that it would formally arrange for the update of the specification after the prototype was proven), induced Company A to breach its contractual obligations with the Systems Integrator to build its software component as specified in the prime contract.

4.5.5.1.3 Possible Claims by Company C

In relation to Company C, additional work was required to re-run the system integration tests after the original tests failed through no fault of its own. The loss suffered by Company C included costs associated with the additional work it was required to perform.

Accordingly, possible claims by Company C include:
- A claim that it was unconscionable for the Systems Integrator to withhold payment and force Company C to re-run the system integration tests when, through no fault of its own it was required to do so to demonstrate that it's component successfully integrated into the full system.
- A claim of economic duress on the part of the Systems Integrator to withhold the progress payment associated with the initial integration test.

The following table summarises the above points:

<table>
<thead>
<tr>
<th>Action</th>
<th>Plaintiff</th>
<th>Defendant</th>
<th>Claim</th>
</tr>
</thead>
<tbody>
<tr>
<td>Misrepresentation (Common Law)</td>
<td>System Integrator</td>
<td>Company A, Company B</td>
<td>Claim is that Company A and B misrepresented to the Systems Integrator that they had the necessary expertise to carry out the work.</td>
</tr>
<tr>
<td>Misrepresentation (Trade Practices Act)</td>
<td>System Integrator</td>
<td>Company A, Company B</td>
<td>Claim is that Company A and B misrepresented to the Systems Integrator that they had the necessary expertise to carry out the work.</td>
</tr>
<tr>
<td>Misrepresentation (Common Law)</td>
<td>Company A</td>
<td>Company B</td>
<td>Claim is that Company B misrepresented to Company A that it would arrange to formally update the interface specification.</td>
</tr>
<tr>
<td>Misrepresentation (Trade Practices Act)</td>
<td>Company A</td>
<td>Company B</td>
<td>Claim is that Company B misrepresented to Company A that it would arrange to formally update the interface specification.</td>
</tr>
<tr>
<td>Breach of Contract</td>
<td>System Integrator</td>
<td>Company A, Company B</td>
<td>Claim is that there was an implied contractual term that Company A and Company B would perform the work with an effective requirements management process in place.</td>
</tr>
<tr>
<td>Professional Negligence</td>
<td>Systems Integrator</td>
<td>Company A, Company B</td>
<td>Claim is that Company A and B were negligent in not putting in place an effective requirements management process.</td>
</tr>
<tr>
<td>Inducing Breach of Contract</td>
<td>Company A</td>
<td>Company B</td>
<td>Claim is that Company B, by stating that it would arrange to formally update the specification, induced Company A to breach its contract with the Systems Integrator by not implementing the contracted interface.</td>
</tr>
<tr>
<td>Estoppel</td>
<td>Company A</td>
<td>Company B</td>
<td>Claim is that Company B should be estopped from denying that it said it would formally update the specification.</td>
</tr>
<tr>
<td>Unconscionable Conduct</td>
<td>Company C</td>
<td>System Integrator</td>
<td>Claim is that it was unconscionable conduct for the System Integrator to withhold payment to Company C and force it to re-run the integration tests.</td>
</tr>
<tr>
<td>Economic Duress</td>
<td>Company C</td>
<td>System Integrator</td>
<td>Claim is that it was economic duress for the System Integrator to withhold payment to Company C and force it to re-run the integration tests.</td>
</tr>
</tbody>
</table>

**Figure 16: Case Study 1 - Summary of Claims**

Some of the claims summarised above naturally carry more weight and have a better opportunity for success than others. Further discussion of these issues is provided below.

**4.5.5.2 Misrepresentation - Common Law**

Chapter 3 above identified that for an action in misrepresentation to be found, four things are needed: A (1) false statement, (2) of fact, (3) directed to the person misled, (4) which was intended to induce the contract.
In the present situation it seems that neither of the individuals from Company A or B made an intentionally false statement. Rather, it is likely that an honest attempt was made by the two employees to produce a better and more cost effective product. Despite this however, as the work was being carried out in a professional environment the issue of negligent misrepresentation is worth considering.

The elements of negligent misrepresentation are that (1) if information or advice is given by a person, and (2) that person represents himself as being competent to give that advice, or (3) realises (or ought to realise) that the other party will rely on the information, and (4) the other party does reasonably rely on that advice to his detriment, then (5) if the advice or information was incorrect the representor will be liable. In the present case study, the employee of Company B said to the employee of Company A:

"I have made some minor changes to the interface which will improve its performance and ease of implementation. If you agree to use this interface then you will see similar benefits. Once we prototype the changes and confirm that they work I will arrange later to update the interface specification formally."

On the facts, it appears that Company A did rely on this statement to implement the new interface design. As a result of such reliance, additional work was eventually required to make the interface conform to the contracted specification.

The false statement made was that Company B was to arrange to formally update the interface specification. Strictly speaking, the statement of intention to update the specification at a later stage would not be sufficient grounds to found an action in misrepresentation unless the promise had become a term of a contract between the two parties. Thus if the statement became a term of the micro-agreement the action may be available. Company A and Company B separately contracted with the Systems Integrator to provide their respective components but, apart from the possible "micro-agreement" formed between the two employees who worked for those companies, no agreement was in place. It could be argued however that the micro-agreement had all necessary elements of a binding contract and as such, the misrepresentation by Company B (that it would arrange to formally update the specification) could conceivably become a term of the micro-agreement.

Despite the above however, if Company B never intended to carry out the actions then it is submitted that, based on the case of Edgington v Fitzmaurice, it may also constitute misrepresentation, even if the false statement did not become a term of the micro-agreement. In the Edgington v Fitzmaurice case it was held that a statement of future intention did amount to a misrepresentation of existing fact because company directors in that case never intended to carry out the statements made.

In relation to remedies, briefly, if misrepresentation is found, damages may be available to Company A through the tort of "deceit", although damages would not be available through contract if the false statement had not become a term of a the micro-agreement (assuming of course that the micro-agreement was a contract). If the statement had become a term then damages under contract may be available. In that instance, the innocent party (Company A) could elect to either rescind the contract for


\[382\] Edgington v Fitzmaurice (1885) 29 Ch.D. 459, per Cotton LJ at 479-480
misrepresentation or sue for damages for breach, but not both\textsuperscript{385}. Regarding the question as to whether damages should be claimed through tort or contract, the tort measure aims at putting the innocent party in the position he had been prior to the tort being committed, while the contract measure examines the position the innocent party would have been in if the contract had been performed.

In relation to the claim by the System Integrator that Companies A and B misrepresented that they had the necessary expertise to carry out the work, it is submitted that, unless it could be shown that either of these companies made a false statement of fact consistent with the other elements of the action, then misrepresentation would not be available. On the facts given in this case study no false statements were made.

4.5.5.3 Misrepresentation under the Trade Practices Act (1974)

Apart from misrepresentation at common law as described above, the *Trade Practices Act (1974)*, Section 52 may also provide some relief to the parties. That Section states that:

"52. (1) A corporation shall not, in trade or commerce, engage in conduct that is misleading or deceptive or is likely to mislead or deceive."

In the first instance, the question is whether this Section is applicable to the facts of the present case study. Generally speaking, it is considered that there is no absolute bar which would preclude bringing an action under Section 52 of the *Trade Practices Act (1974)* in the present case. The reasons for this, as well as other relevant points are made below.

- Although Section 52 comes under *Part V - Consumer Protection*, the Section has not been judicially limited to consumers as defined in Section 4B(1). Specifically, the High Court to date has not set precise boundaries on the applicability of this Section to commercial transactions (see for example *Concrete Constructions (NSW) Pty Ltd v Nelson*)\textsuperscript{384}. Under the facts of the present Case Study therefore Section 52 may be applicable regardless of whether the parties to the contract are "consumers" or otherwise.

- As an aid to interpreting Section 52, Section 51A states *inter alia* that, when representations are made about future matters then they may come within the scope of Section 52:

  "(1) For the purposes of this Division, where a corporation makes a representation with respect to any future matter (including the doing of, or the refusing to do, any act) and the corporation does not have reasonable grounds for making the representation, the representation shall be taken to be misleading."

- The facts of the present case study indicate that the transaction took place in "trade or commerce" when those words are given their natural meaning. In the context of the *Trade Practices Act (1974)* there is also support for the proposition that professional advice can be in "trade or commerce": (See for example *Bond Corporation Pty Ltd v Thiess Contractors Pty Ltd* (1987) 14 FCR 215, per French J.\textsuperscript{385}

\textsuperscript{385} in *Alati v Kruger* (1955) 94 CLR 216 at 222, Dixon C.J., Webb, Kitto, and Taylor J.J. said " on the footing that the contract had been induced by fraudulent misrepresentation the respondent had a choice. He might sue for damages for breach of warranty, for the statement formed one of the terms of the contract and was not only a representation; but he could not do this and rescind for misrepresentation".

\textsuperscript{384} *Concrete Constructions (NSW) Pty Ltd v Nelson* (1990) 169 CLR 594

\textsuperscript{385} *Bond Corporation Pty Ltd v Thiess Contractors Pty Ltd* (1987) 14 FCR 215, per French J
When professional advice is given, whether it be in engineering, computer software development, or other professions, it is often necessary to state opinions or to make predictions as to future events. It was mentioned above that Section 51(A) provides interpretation provisions in relation to future conduct. In addition, a good summary of the law was provided by Toohey J. in James v ANZ Banking Group Ltd 366:

"1. A corporation may be in contravention of s52 whatever its intention or the state of mind of those controlling it...

2. The mere fact that representations as to future conduct or events do not come to pass does not make them misleading or deceptive...

3. Nevertheless, a statement relating to the future may contain an implied statement as to present or past fact. It may represent impliedly that the promisor has a present intention to make good the promise and it may represent impliedly that he has the means to do so...

4. A statement involving the state of mind of the maker of the statement, e.g. Promises predictions and opinions, ordinarily conveys the meaning that the maker of the statement had a particular state of mind when the statement was made and that there was a basis for that state of mind. If the meaning contained in or conveyed by the statement is false in that or any other respect, there will have been a contravention of s52."

In information technology projects it is quite common for predictions to be made about future events such as the expected delivery dates of the completed software. Where an opinion is given or a prediction is made by an expert in the relevant field, then it could be misleading if the expert did not in fact use his or her expertise, or did not otherwise have rational grounds for expressing the opinion or making the prediction: (e.g. Bateman v Slatyer 367).

In order for a plaintiff to succeed in an action under Section 52, it must be shown that there is a sufficient connection between any damage caused and the conduct complained of. 368 In addressing whether there is a sufficient connection, common sense and experience should be taken into account 369. It should also be noted however that it is not necessary for the plaintiff itself to rely on the conduct; thus opening potential claims by third parties. This is important in the context of systems integration contracts generally as potentially many parties may be involved. Some of the parties may have contractual relationships but some may not.

The fact that a person takes a misrepresentation at face value and relies on it without attempting to confirm it by making additional inquiries etc. does not stop that person from making a claim under Section 52 against the person who made the misrepresentation 391.

366 James v ANZ Banking Group Ltd. (1986) 64 ALR 347, at 372
368 e.g. Wardley Australia Ltd. v Western Australia (1992) 175 CLR 514
369 Janssen-Cilag Pty Ltd v Pfizer Pty Ltd (1992) ATPR 41-186.
Overall, it is considered that a claim may be possible under Section 52 of the Trade Practices Act (1974) in the present case study provided (naturally) that the necessary evidence is available. Specifically, the question of whether the System Integrator could make a claim against either Company A or B on the basis that they misrepresented that they had the necessary expertise to manage the development in accordance with the original contract could be difficult depending upon the evidence available. The issues of whether the conduct was misleading or deceptive is a question of fact which needs to be addressed by taking into account all the circumstances of the case. If, for example, either Company A or B made misleading claims that it had significant experience in undertaking similar work it may be possible that those claims would be misleading or deceptive if they did not actually have that expertise.

4.5.5.4 Breach of Contract

4.5.5.4.1 Breach of Implied Terms

The Systems Integrator may claim that both Company A and B were in breach of an implied contractual term that they would put in place an effective requirements management activity. The basis of this claim could be that it is normal industry practice to incorporate such an activity into software development projects to make sure that software requirements are implemented correctly and that changes to these requirements are tracked and updated in a controlled manner.

The facts of the present case study indicate that written formal contracts were in place which covered the work to be performed by both Company A and Company B. It also appears from the facts that no express written terms requiring an effective requirements management activity were in those contracts. Based on the parol evidence rule, any extrinsic evidence of such a term would not be admissible unless one of the exceptions to that rule could be relied upon. Based on one such exception, if it was the intention of the parties that general custom or trade usage terms were to be part of the agreement, but were not specifically incorporated into the written contract, then a court may admit extrinsic evidence to show such usage. Thus if it could be shown that an effective requirements management activity was a customary term used in the trade then it may be possible to imply such a term.

Section 3.4.3 above provided a general discussion of the law in relation to terms implied by custom and in summary:

"It has long been settled that in commercial transactions extrinsic evidence of custom or trade usage is admissible to annex incidents to written contracts in matters with respect to which they are silent... this has been done on the principle or presumption that, in such transactions, the parties did not mean to express in writing the whole of the contract by which they intended to be bound, but a contract with reference to those known usages." 392

Despite the above however it is important to note that:

"It must be so notorious that everybody in the trade enters into a contract with that usage as an implied term. It must be uniform as well as reasonable, and it must have quite as much certainty as the written contract itself." 393

In order to imply such a term into each of the contracts which Company A and Company B had with the Systems Integrator, it would be necessary to show that a significant use of such requirements management techniques were common in the industry to the extent

392 Hutton v Warren [1936] 1 M & W 466, at 475 per Baron Parke.
393 Neilson v Dahl (1879) 12 Ch.D 568, at 575 per Jessel MR.
that they could be regarded as customary. Although ultimately it is a question of fact whether a requirements management activity is customary in the industry, further help in settling the question is provided through the case law as discussed in the following paragraphs.

In respect of the test to be applied to determine whether a term should be implied, in *B.P. Refinery (Westport) Pty Ltd v Shire of Hastings*\(^{394}\) the Privy Council held that, in relation to contracts which are written and which purport to be formal and complete documents:

“For a term to be implied, the following conditions (which may overlap) must be satisfied:

1. it must be reasonable and equitable;
2. it must be necessary to give business efficacy to the contract, so that no term will be implied if the contract is effective without it;
3. it must be so obvious that 'it goes without saying';
4. it must be capable of clear expression;
5. it must not contradict any express term of the contract.”

This passage has been accepted and applied on many occasions by the High Court\(^{395}\) and thus carries significant weight. It is also important to note that the above test does not apply if the contract is not recorded in writing in a formal sense. In such a case the test as to whether a term may be implied into a contract is whether the implication of the term is “necessary for the reasonable and effective operation of a contract of that nature in the circumstances of the case”\(^{396}\).

The passage cited above *B.P. Refinery (Westport) Pty Ltd v Shire of Hastings* identifies the elements necessary to imply terms generally. More specifically, in relation to terms implied by custom, in *Con-Stan Industries of Australia Pty. Ltd. v Norwich Winterthur Insurance (Australia) Limited*\(^{397}\), the High Court summarised the elements required to allow the implication of terms into a contract on the basis of custom. Specifically, the Court expounded the following propositions:

“(1) The existence of a custom or usage that will justify the implication of a term into a contract is a question of fact *(Nelson v Dahl* (1879) 12 Ch.D. 568, 575).

(2) There must be evidence that the custom relied on is so well known and acquires in that everyone making a contract in that situation can reasonably be presumed to have imported that term into the contract. ... However, it is not necessary that the custom be universally accepted, for such a requirement would always be universally defeated by the denial by one litigant of the very matter that the other party seeks to prove in the proceedings.

(3) A term will not be implied into a contract on the basis of custom where it is contrary to the express terms of the agreement.

(4) A person may be bound by a custom notwithstanding the fact that he had no knowledge of it.”

\(^{394}\) *B.P. Refinery (Westport) Pty Ltd v Shire of Hastings* (1977) 16 ALR 363 at 365


\(^{396}\) *Hawkins v Clayton* (1988) 164 CLR 539 at 573

Overall it is considered that there is some scope for the implication of a term into a systems integration contract to the effect that a requirements management function should be in place. Certainly if the supplier claims to have a Software Quality Management System in place which is based on some of the common industry standards\(^{398}\), then it is conceivable that it was the intention of the parties to the contract that management control over requirements would be in place.

Conversely however it is also generally recognised that it may cost more and take more time to fully implement a formal requirements management function and that in some cases it may not be appropriate. The degree of formality would probably depend on the scope and size of the project, the extent to which the requirements were known at the time of contract, and the amount of change expected in the software in future. Overall however it is well recognised within the software industry that management of requirements can be critical to the success or otherwise of a large software development.

4.5.5.4.2 The Micro-Agreement

Focusing on the micro-agreement itself, the two employees from Company A and Company B together agreed to change their implementation of the interface specification with the aim of enhancing the product and simplifying the design. In order to determine what legal remedies are available to the various parties one issue is whether this agreement constituted a binding contract in the traditional sense.

4.5.5.4.3 Does the micro-agreement constitute a contract?

It seems reasonable to argue that the micro-agreement did constitute a binding contract (depending on some facts which are not identified above such as the actual conversations and documents passed between the parties during negotiation of the micro-agreement). Presumably the employees, in acting as agents for their respective companies had the legal capacity and intention to create a legal relationship. It is also likely that evidence would be available to show that an offer and acceptance process took place during negotiations between the two. Consideration could possibly be argued on the basis that the "value" derived would be the potential savings in time and cost if the revised interface was implemented instead of the originally specified interface. In order to make this claim however it must be shown (among other things) that the consideration was firstly, "sufficient" enough for a binding contract to be formed, and not merely "illusory"\(^{399}\); and secondly, that the consideration moved from the promisee\(^{400}\).

In relation to the first point (sufficiency) it is possible that some quantitative estimate of the value could be derived, for example, by calculating the time saved by implementing the "new" interface and then applying an hourly rate. This method could provide some evidence of "sufficiency" of consideration (the "adequacy" itself being effectively irrelevant for the purposes of consideration\(^{401}\).

In relation to the issue of whether the consideration moved from the promisee, if Company B promised Company A (expressly or impliedly) that Company A’s costs would be reduced by implementing the revised interface, then it may be possible to argue that that

\(^{398}\) such as ISO/IEC 12207 Information Technology - Software Life-cycle Processes, 1995-08-01; IEEE STD 1298/A3563.1, Software Quality Management System

\(^{399}\) for example: White v Blutt (1853) 23 LJ (NS) Ex 36

\(^{400}\) see for example: Couls v Bagot’s Executor and Trustee Co Ltd (1967) CLR 460

\(^{401}\) Consideration needs to be sufficient but need not be ‘adequate’: see for example: Barba v Fule Corp of Victoria (1976) 136 CLR 120
promise itself is valuable consideration (provided that a court, if called upon to do so, could enforce that promise)\textsuperscript{402}.

Possible arguments against the view that this micro-agreement constituted a binding contract could include whether the parties were operating under a mistake of fact in that neither individual realised that their actions in changing the interface would significantly impact third parties. As these individuals were employees, it is reasonably clear that the parties to this micro-agreement (i.e. Company A and B) would not be operating under this mistake because each of their contracts with the Systems Integrator explicitly called for conformance with the specification.

4.5.5.4 What is the effect of the micro-agreement in respect of contract?

If it could be shown that the micro-agreement constituted a binding contract then the legal result would be the remedies normally associated with such contracts. These remedies could possibly include (depending upon the facts) for Company A: rescission, damages for breach, or specific performance of the promise by Company B that costs would be reduced by implementing the new interface; and for Company B: rescission, damages for breach, or specific performance of the “promise” by Company A that it would implement the revised interface.

4.5.5.5 Estoppel

Company A may claim that Company B should be estopped from denying that it said that it would formally arrange to update the interface specification. What is the basis of this claim and if available, what effect would it have?

4.5.5.5.1 The Argument for Estoppel

The basic principle of estoppel is that “the law should not permit an unjust departure by a party from an assumption which he has caused another party to adopt or accept for the purpose of their legal actions. The departure will be unjust if the other person having acted or abstained from acting on the basis of the assumption, would suffer a detriment if the opposite party were afterwards allowed to set up rights against him inconsistent with that assumption.”\textsuperscript{403}

Based on the *Waltons v Maher*\textsuperscript{404} decision, commentators including Elliott\textsuperscript{405} have attempted to summarise the practical effects of the decision as follows:

- “Practitioners must now be aware that conduct and statements during the course of pre-contractual negotiations can lead to binding obligations and an award of damages, even if the parties have not entered into a contract”\textsuperscript{406};

- “Practitioners should look at the total circumstances and the actions of both parties, where the one party has acted to his detriment on an assumption caused by the statements, conduct, advice, or failure to advise or act of another party. The relationship of the parties is important. If there is some form of reliance by one party on the other; if there is some form of broad duty cast upon one party to give true advice, or not to allow the other party to continue to act, then in these circumstances the equitable notion of unconscionability or fairness...”\textsuperscript{406}


\textsuperscript{404} *Walton's (Interstate)* v Maher & Anor. (1988) 62 ALJR 110

\textsuperscript{405} see Elliott, Paul: "Developments in Equitable Estoppel since Waltons Stores", being a paper delivered at the BLEC Workshop: "The Law of Contract Disputes, 1994"

\textsuperscript{406} Ibid

Paul P. Parnell

106
will apply, and one party will be estopped or prevented from going back on the state of affairs he had induced another to accept or act upon.\footnote{ibid}

- In order to claim estoppel it is reasonably clear that "there must be some sense of inherent injustice, of clear unfairness existing in the circumstances."\footnote{ibid}

Overall, Elliot summed up the position in relation to estoppel as follows:

"The doctrine of estoppel as espoused by Walton's Stores has had a profound effect on the law of contract. Promises not supported by consideration may now be enforceable. Pre-contractual negotiations may become binding. The relief granted pursuant to an estoppel can be equivalent to the relief granted as if a contract existed. Application of the Statute of Frauds and requirements of writing may have no application in the areas of estoppel."\footnote{ibid}

The situation presented in this case study obviously constitutes a commercial arrangement and the question arises as to whether estoppel is applicable in such circumstances. Courts have at times been reluctant to apply the principle of equitable estoppel to commercial transactions, particularly if the parties to the contract have relatively equal bargaining power. For example, in \textit{Austotel Limited v Franklins Limited}\footnote{Austotel Limited v Franklins Limited (1989) 16 NSWLR 582 at 585} it was held that:

"We are not dealing here with ordinary individuals invoking the protection of equity from the unconscionable operation of a rigid rule of the common law. Nor are we dealing with parties which were unequal in bargaining power. Nor were the parties lacking in advice either of a legal character or of technical expertise... at least in the circumstances such as the present, courts should be careful to conserve relief so that they do not, in commercial matters, substitute loyalty conscience for the hard headed decisions of business people."

In addition to the above, other concerns about applying estoppel in commercial relationships have been expressed in the following terms:

"What is clear from the decisions is that in principle the doctrine of estoppel can be applied to pre-contractual negotiations. However it is clear that the circumstances of the negotiation will be examined in detail by a Court. If in general there is equality of the parties and the parties have made conscious decisions not to legally bind themselves based on commercial factors, it may be that a Court will be reluctant to apply the principle of unconscionability in these circumstances. The problems in applying the Walton's type doctrine of estoppel to pre-contractual negotiations was clearly outlined by Kirby P. In the case of \textit{State Rail Authority of New South Wales v Heath Outdoor Pty. Ltd.} (1986) 7 NSWLR 170 at 176 where he said as follows:

'Too great a willingness by the Courts to discern, in pre contract negotiations, a basis for estoppel will have the effect of introducing a serious element of uncertainty into our law of contract. It may also encourage expensive litigation in which the terms of the writing are put to one side and the courts busily engaged (as we have been) in a minute examination of the wilderness of pre contract conversations. This may be the reason, at least in the case of written contracts which are accepted by the parties and are not varied or elaborated, to hold the parties to the applicable terms of such contracts and to limit carefully the development of the law of estoppel, least it seriously undermine the adherence to bargains which are such an important feature of modern economic life. But even if

\footnote{ibid}
\footnote{ibid}
\footnote{Elliott, Paul: "Developments in Equitable Estoppel since Waltons Stores", being a paper delivered at the BLEC Workshop: "The Law of Contract Disputes, 1994". p111.}
estoppel is pushed so, to tender sense of the unconscionable may mislead the courts into substituting the court's notions of equity as between contracting parties for the bargain which the parties have negotiated and accepted.411

The above passage indicates that it may be difficult for any claim of estoppel to be made where the parties are on equal footing in a commercial transaction. Overall however it appears from the cases that there is some scope for estoppel to apply if there is "some sense of inherent injustice, of clear unfairness existing in the circumstances".412 On the facts given in this case study, it is by no means certain that a claim of estoppel would be found however it appears that the circumstances are such that unfairness and injustice may arise because of the actions of Company B. Specifically, it is submitted that it could be argued that Company B, by indicating that it would formally arrange to update the interface specification formally at a later stage, induced Company A to act to its detriment by relying on Company B's statement and implementing the modified interface.

Regarding the "detriment" suffered, one issue is whether the loss suffered by Company A constitutes a "material detriment" for the purposes of estoppel:

"But what makes it unjust to permit the departure from an assumption so induced is that, were it permitted, the party so induced would through making the assumption find himself in a position occasioning material detriment to himself. Without this element there is no estoppel"413.

The question is therefore whether the greater expense caused to Company A as a result of the actions of Company B is sufficient to constitute a material detriment for the purposes of establishing estoppel:

"In order for there to be a detriment sufficient to found an estoppel, the party alleging the estoppel must show that the detriment is more than a mere possibility; it must be a real or material detriment, although pecuniary loss may not be necessary."414

Ultimately it is likely to be question on the facts but irrespective of the above, any remedy must be proportional to the detriment:

"A central element of that doctrine is that there must be a proportionality between the remedy and the detriment which is its purpose to avoid. It would be wholly inequitable and unjust to insist upon a disproportionate making good of the relevant assumption"415.

Further discussion of remedies is provided below.

4.5.5.5.2 Remedies
If an action for estoppel is made out in this case study, what is the practical effect?

In the past, the type of estoppel which was found would to some extent determine the remedies available. Specifically, "common law" estoppel would provide remedies through normal contract remedies (because the pre-contractual representations become part of

412 see Elliott, Paul: "Developments in Equitable Estoppel since Waltons Stores", being a paper delivered at the BLEC Workshop: "The Law of Contract Disputes, 1994".
413 Newbon v City Mutual Life Assurance ltd (1935) 52C.L.R.723 at 734 as cited in In Re: Anthony Chin; Peter Desmond O'Shea; Chin Sik Hoe; Capricornia Motels Pty Ltd and Capricornia Penthouse Restaurants Pty Ltd v Jeffrey George Miller and Lynette Nona Miller (1981) 56 FLR 359 No. 4 of 1980
415 The Commonwealth v. Verwayen (1990) 170 CLR 394 F.C. 90/036, at paragraph 36, per Mason CJ
the contract), whereas “equitable” estoppel would provide a set of rights which were separate to any contractual right, and thus a separate set of remedies. The decision in \textit{Waltons v Maher}\textsuperscript{416} focused on the separate areas of common law and equitable estoppel, however more recent cases have (to some extent) merged these doctrines. In \textit{Foran v Wight}\textsuperscript{417}, Mason CJ found that:

"On further reflection it seems to me that we should now recognise that a common law estoppel may arise out of a representation or mistaken assumption as to future conduct."

The effect of this statement is that representations as to future conduct, (as opposed to representations as to existing fact), may be the subject of common law estoppel as well as equitable estoppel. Mason further elaborated in \textit{Commonwealth v Verwayen}\textsuperscript{418}:

"In conformity with the fundamental purpose of all estoppels to afford protection against the detriment which would flow from a party’s change of position if the assumption that led to it were deserted, these developments have brought a greater underlying unity to the various categories of estoppel. Indeed, the consistent trend in the modern decisions points inexorably towards the emergence of one overarching doctrine of estoppel rather than a series of independent rules."

So what remedies are available? In \textit{Commonwealth v Verwayen}\textsuperscript{419}, Mason C.J. discussed remedies in the following terms:

"So, in \textit{Walton’s Stores}, a majority of this court concluded that equitable estoppel entitled a party only to that relief which was necessary to prevent unconscionable conduct and to do justice between the parties. Mason C.J. and Wilson J. referred to the statement of Scarman L.J. in \textit{Crabb v Aron District Council} (1976) Ch. 179 at 198, that the court should determine what was ‘the minimum equity to do justice to the plaintiff’. We went on to state: ‘holding the representor to his representation is merely one way of doing justice between the parties’... it follows, as a matter of principle and authority, equitable estoppel will permit a court to do what is required in order to avoid detriment to the party who has relied on the assumption induced by the party estopped, but no more. In appropriate cases that will require that the party estopped be held to the assumption created, even if that means the effective enforcement of a voluntary promise. To that extent there is an overlap between equitable estoppel generally and estoppel be conduct in its traditional form. But since the function of equitable estoppel has expanded and it has become recognised that an assumption that as to future fact may ground an estoppel by conduct at common law as well as in equity, it is anomalous and potentially unjust to allow the two doctrines to inhabit the same territory yet produce different results."

In relation to the present case study the practical effect is therefore that, despite the representations being made as to future conduct, a court may require that the party estopped (Company B) be held to the assumption created (that Company B would arrange for the interface specification to be formally updated and re-issued), even if that means the effective enforcement of a voluntary promise. Overall it is submitted that, if an action in estoppel is found against Company B, then remedies such as specific performance and possibly damages may be available to Company A as a result.

\textbf{4.5.5.6 Economic Duress}

In the facts of the present case study, the System Integrator withheld payment to Company C because the integration testing failed (although as subsequently discovered, through no fault of Company C), and then “forced” Company C to re-run the integration.

\textsuperscript{416} Waltons (Interstate) v Maher & Anor. (1988) 62 ALJR 110
\textsuperscript{417} Foran v Wight (1990) 64 ALJR 1
\textsuperscript{418} Commonwealth v Verwayen (1990) 64 ALJR 540 at 545
\textsuperscript{419} Commonwealth v Verwayen (1990) 64 ALJR 540 at 545
tests before payment was made. Does this constitute economic duress on the part of the Systems Integrator?

In relation to economic duress, where one party makes threats against another party in relation to economic interests, and that duress influences the formation of a contract, then a claim of economic duress may be available. In *Universe Tankships* 420, Lord Scarman listed the elements required for economic duress as:

- firstly, pressure which amounts to a compulsion of the will of the party coerced; and
- secondly, the illegitimacy of that pressure.

Based on the above definition of the action, even if there was “duress” it did not appear to influence the formation of a contract because the contract between the Systems Integrator and Company C was already in place (and in fact the contract was approaching completion). Consequently the action for economic duress is unlikely to be available in the present circumstances. Despite this, in order to explore the situation fully given slightly different fact situations, the following points are made.

If for example the Systems Integrator did not withhold payment, but rather threatened to withhold future work from Company C, then this may be an example of where the duress influenced the formation of a contract. Accordingly, under this slightly modified fact situation, it appears that “pressure” was applied by the Systems Integrator to Company C to re-run the integration tests, however whether this pressure was significant enough to amount to a “compulsion of will” is arguable on the facts. In *North Ocean Shipping* 421, it was regarded that the first test was satisfied if there existed no reasonable alternative for the party subject to the duress. Under the present circumstances, it is submitted that a “reasonable alternative” for Company C may have been to take action against the Systems Integrator via breach of contract (i.e. for not making the payment after the integration testing was performed). In addition, in *Crescendo Management Pty. Ltd. v Westpac Banking Corp.* 422 it was implied that the reasonable alternative test may not carry so much weight relative to the second element.

Regarding the second element identified by Lord Scarman above, although the pressure in *North Ocean Shipping* was unlawful, it was not suggested that unlawful pressure is necessary, although it appears that, as for duress to goods, the pressure must be of a serious nature (*Atlas Express Ltd. v Kefco (Importers & Distributors) Ltd.* 423). It would thus appear that, regardless of whether the pressure applied by the Systems Integrator was illegal or not, provided the pressure was illegitimate and significant enough the action may be available.

In relation to remedies, if economic duress is found, the innocent party may treat the contract as void, although as seen in *North Ocean Shipping*, this right can be lost through affirmation.

### 4.5.5.7 Unconscionable Transactions

As discussed in Section 3.2.6.2, in Australia the *Trade Practices Act, 1974* provides some statutory requirements related to unconscionability, namely, under Sections 51AA, 51AB and 51AC.

---

420 *Universe Tankships Inc. v International Transport Workers Federation* [1983] 1 A.C. 336
421 *North Ocean Shipping Co Ltd v Hyundai Construction Co Ltd (The Atlantic Baron)* [1979] Q.B. 705
422 *Crescendo Management Pty. Ltd. v Westpac Banking Corp.* (1988) 19 N.S.W.L.R. 40 per McHugh J.
It is submitted that a claim under Sections 51AA and 51AB would not succeed in the present circumstances. That Section states that:

"51AA(1) A corporation must not, in trade or commerce, engage in conduct that is unconscionable within the meaning of the unwritten law"

The main problems with Section 51AA in the present context are that firstly, Section 51AA requires that there be a position of "special disadvantage" in the weaker party, and secondly, Section 51AB applies to "goods or services of a kind ordinarily acquired for personal, domestic or household use or consumption".\(^{424}\) It is submitted that on the facts it would be difficult to show that Company A was in a position of "special disadvantage" in the traditional sense. Both parties seemingly entered into a commercial arrangement on equal footing. In addition, the nature of the software being developed in this case study is unlikely to be classified as "goods or services of a kind ordinarily acquired for personal, domestic or household use or consumption", bearing in mind comments made in Section 3.2.6.2 above.

In respect of Section 51AC however there may be some relief available. We have seen in the present case study how the Systems Integrator refused to provide payment to Company C for integration testing. Section 51AC provides that (inter alia):

**51AC Unconscionable conduct in business transactions**

1. A corporation must not, in trade or commerce, in connection with:
   - the supply or possible supply of goods or services to a person (other than a listed public company); or
   - the acquisition or possible acquisition of goods or services from a person (other than a listed public company);

engage in conduct that is, in all the circumstances, unconscionable.

Could a claim be made against the Systems Integrator under this Section (specifically, Section 51AC(1)(b) in that it engaged in unconscionable conduct in connection with the acquisition of services from Company C by refusing to provide payment for integration testing? Section 51AC also lists a number of issues which the court can take into account when deciding the issue. Specifically, these issues are listed in Section 51AC(4):

1. the relative strengths of the bargaining positions of the acquirer and the small business supplier; and
2. whether, as a result of conduct engaged in by the acquirer, the small business supplier was required to comply with conditions that were not reasonably necessary for the protection of the legitimate interests of the acquirer; and
3. whether the small business supplier was able to understand any documents relating to the acquisition or possible acquisition of the goods or services; and
4. whether any undue influence or pressure was exerted on, or any unfair tactics were used against, the small business supplier or a person acting on behalf of the small business supplier by the acquirer or a person acting on behalf of the acquirer in relation to the acquisition or possible acquisition of the goods or services; and
5. the amount for which, and the circumstances in which, the small business supplier could have supplied identical or equivalent goods or services to a person other than the acquirer; and

\(^{424}\) *Trade Practices Act, 1974 Section 51AB(5)*
(f) the extent to which the acquirer's conduct towards the small business supplier was consistent with the acquirer's conduct in similar transactions between the acquirer and other like small business suppliers; and

(g) the requirements of any applicable industry code; and

(h) the requirements of any other industry code, if the small business supplier acted on the reasonable belief that the acquirer would comply with that code; and

(i) the extent to which the acquirer unreasonably failed to disclose to the small business supplier:
   (i) any intended conduct of the acquirer that might affect the interests of the small business supplier; and
   (ii) any risks to the small business supplier arising from the acquirer's intended conduct (being risks that the acquirer should have foreseen would not be apparent to the small business supplier); and

(j) the extent to which the acquirer was willing to negotiate the terms and conditions of any contract for the acquisition of the goods and services with the small business supplier; and

(k) the extent to which the acquirer and the small business supplier acted in good faith.

Section 51AC has not been judicially considered however it is submitted that the above list of issues is comprehensive enough to provide relief under the present factual situation. Specifically, it could be claimed by Company C that undue influence or pressure was exerted on it, and unfair tactics were used against it by the Systems Integrator in that it withheld money and refused to make a progress payment for work which was completed in accordance with the contract (s51AC(4)(d)).

Although the courts have been somewhat reluctant to apply the concepts of unconscionability in purely commercial settings, it appears that Section 51AC is an attempt by Parliament to change this situation.

4.5.6 Suggested Improvements in the Law

4.5.6.1 Introduction

It has been seen in the above case study that various remedies may be available to the parties to this micro-agreement under existing law. Despite this it is important to note that the micro-agreement itself appears to constitute a special kind of interaction between the parties which is not necessarily formalised to any great extent, but nonetheless may have a significant effect on the outcome of the project.

In order to minimise the problems associated with such interaction and maximise the benefits, it is submitted that legal mechanisms which encourage co-operation between the parties and which prevent unfairness are the most useful.

4.5.6.2 Implied Contractual Duty to Co-operate

As discussed in Chapter 3 Contract Interpretation and Related Issues above, sometimes the law imposes an implied duty to co-operate on the parties to a contract. This case study focuses on requirements management and to that end, in an information technology development or systems integration contract, the party contracted to provide the system may also be required to undertake systems analysis work to establish the overall requirements of the system before it is implemented. If the buyer of the system does not co-operate with the developer to provide relevant information about the requirements then the buyer itself may be in breach of an implied duty to co-operate.
The degree to which the general law supports this concept is however limited. In *Mona Oil Equipment & Supply Co. Ltd. v Rhodesia Railways Ltd.* [425] it was stated:

"It is, no doubt, true that every business contract depends for its smooth working on co-operation, but in the ordinary business contract, apart, of course, from express terms, the law can enforce co-operation only in a limited degree - to the extent that it is necessary to make the contract workable. For any higher degree of co-operation the parties must rely on the desire that both of them usually have that the business should get done."

As communication and co-operation between the parties are likely to be key elements in the successful development of a large information technology contracts, it is submitted that the duty to co-operate should, as much as possible, be enforceable. Although the common law doctrine applies "to the extent that it is necessary to make the contract workable", it would be preferable if this concept could be strengthened without over burdening the parties to the contract unnecessarily.

As discussed in Chapter 2 above, the resolution of technical issues early in the development life cycle is critical, particularly in relation to requirements and in detection of defects. As such, it is crucial that any co-operation occur not only after things have gone wrong, but more importantly, from the early stages of the contract. As such, it is submitted that the doctrine of an "implied duty to co-operate" should be modified to explicitly apply "throughout the period of the contract, to the extent that it is necessary to make the contract workable". Alternatively, this concept should be built into the contract directly.

Overall, it is submitted that the focus of the duty to co-operate would then be more consistent with the time it is most required - in the early lifecycle phases of a large information technology contract. Although the duty as it currently stands does not of course preclude that, it is likely that early in the contract each of the parties will be heavily involved in meeting their own contractual obligations and as such may be less reluctant to spend the time and effort necessary to effectively interact with the other parties. This modification would help to avoid that.

Thus it is recommended that the contractual doctrine of an "implied duty to co-operate" be modified to explicitly apply "throughout the period of the contract, to the extent that it is necessary to make the contract workable".

### 4.5.7 Industry Code of Conduct

A discussion of a proposed industry code of conduct: *Information Technology Systems Integration and Outsourcing Code of Conduct* is provided in Section 4.11. The discussion provided in that Section addresses issues and makes recommendations for the disclosure of requirements management practices in the information technology systems integration and outsourcing industries.

### 4.5.8 Findings

In relation to this Case Study key findings are summarised below.

---

### Key Findings

**Finding 1:** A micro-agreement existed in this case study. (Specifically, the micro-agreement consisted of an agreement between employees of two companies to implement a slightly modified interface rather than the one specified in the prime contract).

The micro-agreement in this case study consisted of an informal oral agreement between an employee of Company A and an employee of Company B to implement a slightly modified interface rather than the one contractually specified. Thus, the micro-agreement consisted of an informal oral interaction between two or more individuals or bodies associated with a large information technology development.

**Finding 2:** The micro-agreement had a significant impact on the development. (Specifically, the end result of the micro-agreement was that three of the sub-contractors needed to perform additional work at increased cost and schedule).

The impact of the micro-agreement in this case study was that (among other things) all three sub-contractors needed to perform additional work (with associated cost and schedule claims). This was despite the fact that if the micro-agreement was put into effect (i.e. with the interface modifications), the final product would have been technically superior and the integration activities simpler for all parties. Note that the micro-agreement had a significant effect on the outcome of the project despite the formalities of any contracts that were in place at the start of the development.

**Finding 3:** The micro-agreement was highly likely to occur during the conduct of the development because of the inherent nature and size of the project and the lack of formal Requirements Management activity.

With the many parties involved in this development it was essential that effective communication occurred to resolve technical and managerial issues. In larger software developments where there can be a large amount of interaction, micro-agreements are likely to occur.

**Finding 4:** The micro-agreement was an important form of communication between the parties, and would have resulted in a product of higher quality with reduced cost and schedule. Accordingly, micro-agreements should not be prevented from occurring but rather allowed to occur under controlled conditions.

Physical, managerial, or even contractual "roadblocks" should not be put in place with the aim of minimising micro-agreements. Communication is an essential tool of any large software development activity. The aim should be rather to provide an environment where a degree of control is in place to ensure that micro-agreements do not introduce significant risk, and that the potential benefits from them are realised.
| Finding 5: | The information technology agreement in this case study did not adequately take the effect of the micro-agreement into account. |

It appears that there was no effective contractual mechanism for the management of requirements.

| Finding 6: | The project needed to be carefully managed to minimise risk and maximise benefits of the micro-agreement. |

| Finding 7: | Where an opinion is given or a prediction is made by an expert in the relevant field, then it could be misleading if the expert did not in fact use his or her expertise, or did not otherwise have rational grounds for expressing the opinion or making the prediction. |

| Finding 8: | Conduct and statements made during the course of pre-contractual negotiations can lead to binding obligations and an award of damages, even if the parties have not entered into a contract. |

| Finding 9: | Information technology custom or trade usage may be implied into a contract. Custom or trade usage within the IT industry may include expectations of having fundamental software engineering practices in place such as quality assurance systems. |

Recommendations on how to minimise the risk and maximise the benefits of micro-agreements are provided in the following Section.

### 4.5.9 Recommendations

In line with these findings it is submitted that the best way of dealing with micro-agreements (from a legal perspective) is to use legal devices which are designed to encourage interaction and innovation between the parties. These devices should be made available through general law or enforced contractually. Specifically, it is recommended that:

#### Recommendation 1: Link the contract to Software Engineering “best practice”.

Ensuring that a development is undertaken using software engineering best practice is important. What exactly “best practice” is depends on the particular development and the technologies used. The following recommendation gives an example of one such best practice which should be undertaken in a formal sense in any significant systems integration or outsourcing project.
Recommendation 2: Provide support for an effective Requirements Management Process in the Contract.

It is a demonstrated fact that over the life of a project, the system and software requirements will change. It is submitted that an effective mechanism for coping with such change is crucial to the successful outcome of a project. Linking this activity to the contract will provide additional weight to it to ensure that the activities are undertaken.

Recommendation 3: Micro-agreements should not be prevented from occurring but rather allowed to occur under controlled conditions.

The risks and benefits of micro-agreements need to be managed. As both the potential benefits and risks can be significant, the key is to manage micro-agreements effectively rather than prevent them from occurring altogether.

Recommendation 4: The contractual doctrine of an “implied duty to co-operate” should be modified to explicitly apply “throughout the period of the contract, to the extent that it is necessary to make the contract workable”.

In any contract it is difficult to foresee and take into account every possible contingency and systems integration and outsourcing contracts are no exception. Co-operation between the parties is crucial to large software projects for reasons previously discussed and accordingly modification of the contractual doctrine of an implied duty to co-operate in line with the above recommendation may help co-operation throughout the lifecycle of the development.

Recommendation 5: As a strategy for avoiding a potential action under Section 52 of the Trade Practices Act, 1974 IT practitioners should be made aware that where an opinion is given or a prediction is made by an expert in the relevant field, then it could be misleading if the expert did not in fact use his or her expertise, or did not otherwise have rational grounds for expressing the opinion or making the prediction.

Recommendation 6: As a strategy for avoiding a potential action under the principals of estoppel, IT practitioners should be made aware that conduct and statements made during the course of pre-contractual negotiations can lead to binding obligations and an award of damages and other remedies, even if the parties have not entered into a contract.
Recommendation 7: As a strategy for avoiding any unintended consequences due to terms being implied into a contract, IT practitioners should be made aware that terms related to custom or trade usage may be implied into a contract. Custom or trade usage within the IT industry may include expectations of having fundamental software engineering practices in place such as quality assurance systems. Where there is any doubt terms should be expressed in contracts.

Recommendation 8: A voluntary “Industry Code of Conduct” should be developed in accordance with the Trade Practices Act, 1974 Part IVB to ensure a Systems Integrator or Outsourcing organisation discloses to the customer the Requirements Management activities to be undertaken.

The key concept here is disclosure. If a full and frank disclosure of the requirements management activities to be undertaken by the systems integrator or outsourcer is made, then all parties should be in a better position to decide on the terms and scope of any subsequent contract.

4.5.10 Conclusion

One of the fundamental issues in this case study is about ensuring effective communication between the parties. Traditional legal mechanisms are available to “clean up” the adverse effects of disputes however it is submitted that the key to minimising the problems and maximising the benefits of micro-agreements is to encourage controlled interaction between all parties to the development. To some extent this idea is summed up as follows:

“The law of remedies fails to recognise the importance of continuing relationships in business. If a trader wishes to continue with a relationship, it does not sue its partner nor even implicitly or explicitly threaten to do so. Even when a relationship breaks down, litigation is discouraged by the possibility of obtaining an adverse business reputation of being litigious. Resort to the courts is also expensive in direct terms, and in the time taken from senior executives.”

This case study focused on a set of facts which was designed to highlight problems with requirements management. A micro-agreement was described which highlighted the problems which could arise in respect of a fundamental software engineering activity. A discussion was provided which aimed to firstly, show how the problems could have been avoided in the in the first place; and secondly, if they couldn’t be avoided, what relief the law could provide.

Overall, this case study has shown that micro-agreements can have a major impact on the outcome of information technology projects.

4.6 CASE STUDY 2 - MICRO-AGREEMENTS IN THE CONTEXT OF SOFTWARE PROJECT PLANNING

4.6.1 Introduction and Explanation of the Activity

Chapter 2 of this document described the "life-cycle" phases which may be applied to the development and integration of software based system. These phases are often used to plan activities with the aim of meeting overall business objectives, within a defined timescale and at a defined cost. As discussed in Chapter 2, it can be difficult to estimate cost and schedule associated with even a fairly straightforward software project let alone some of the larger developments. Large systems integration projects which combine software and hardware components from a number of manufacturers provide their own problems in relation to predicting cost and schedule. The software project planning activity is intended to provide an ongoing estimate of the work. Specifically, it has been described as:

"Software Project Planning involves developing estimates for the work to be performed, establishing the necessary commitments, and defining the plan to perform the work. A plan is established to address the commitments to the customer according to the resources, constraints, and capabilities of the project. The plan provides the basis for initiating the software effort and managing the progress of the work."\(^{227}\)

4.6.2 Example of a Micro-agreement

The following is a scenario which could occur in the context of a large systems integration contract.

A contract for the development of a large military system specified that the work was to be conducted in accordance with a technical standard\(^{228}\) for software development and documentation. Although the standard used is capable of providing support for incremental or evolutionary development, to the extent that multiple software builds can be evolved and delivered over time, the particular contract specified a basic waterfall lifecycle\(^{229}\) which involved finalising products and documentation for each phase before moving onto the next phase.

Specific milestone dates were identified in the contract for the completion of particular documents. Specifically, the contract required that a "Software Requirements Specification" and "Software Design Description" be completed by the developer of each sub-system and approved before the software coding phase was to be commenced. Due to the large size of the overall system, a number of critical sub-systems were completed first which left one last sub-system to be integrated in a very short timeframe. This sub-system was technically complex as it was to be used to control missile launch and guidance parameters as well as control the application of electrical power to a missile. Due to the complex nature of the software, its algorithms and timing relationships, attempts to precisely define all aspects of the system requirements and design in the specified documents proved almost impossible without prototyping (for a discussion of prototyping see Section: 2.5.5.3)


\(^{228}\) A number of software standards have been published of which MIL-STD-498 "Software Development and Documentation", 5 Dec 1994, US DOD is an example.

\(^{229}\) See Section 2.4.2.1 for a discussion of the waterfall lifecycle model.

Paul P. Parnell 118
Due to schedule pressure a micro-agreement was made between the Systems Integrator and the Developer of the sub-system to implement the sub-system in accordance with draft versions of the System Requirements Specification and System Design Specification which had been produced by the Developer and delivered as required under the contract. These documents were not fully adequate at that stage because of the complexity involved in the sub-system. They were delivered however in order to meet the contractual obligations of the Developer in relation to delivery of those documents by a specific time.

The intention of the parties was that once a working prototype was available, the sub-system software and documentation were to be “fine tuned” to correctly reflect measurements taken in the laboratory. Extensive “informal” (or non-contractually required) testing in the laboratory was conducted and the results documented.

Incomplete specifications were thus delivered in order to meet the contractual deadlines when it was understood that those documents would need to be revised and re-issued in the future.

The micro-agreement made was thus to explicitly modify the contractual “lifecycle” process as defined in the main contract and build a working software prototype before the contracted requirements and design documentation were complete. The micro-agreement thus was integral to the planning activity but went against the contracted lifecycle process. The final product, although technically adequate, did not match up with the formally issued documentation referred to above. Unplanned updates to the specifications and test procedures were thus required to match with the developed system and as a consequence the formal testing of the software could not be completed in accordance with the scheduled contracted milestones.

4.6.3 Potential Problems arising from the Micro-agreement

As a result of this micro-agreement:

1. Although the product itself was produced within the desired timeframe, the requirements and design documentation did not fully reflect the software actually in the system. This had a flow on effect that formal testing could not be conducted as scheduled because the documentation was not complete. As the functionality of the software was related to missile launch and was thus safety critical, the contractual requirements for validation which were specified in the original contract could not be bypassed (nor should they have been) and accordingly additional time and effort was needed to complete the test and validation work.

2. The product was tested and delivered late.

3. A further revision of the specification documents was required which was not spelt out in the original contract. These documentation updates were produced by the developer.

4.6.4 Avoiding the Problem

One of the main causes of the problems identified in this case study is probably inadequate software development planning. If a Software Development Plan was produced for the project then it was probably lacking both in terms of its initial content as well as its lack of flexibility to cope with change.

It is acknowledged that specifying milestone dates in the contract for critical documents (such as the Software Requirements Specification) is beneficial in providing certainty to the parties. In this case however, the problem appeared to be that there was no
recognised plan put in place to cope with the changes. In this particular example, fortunately the customer was able to accommodate the late delivery and the suppliers were able to cope with the additional work associated with issuing revised documentation sets.

If the parties had however strictly adhered to the contract then it is likely that either the product would not have been technically adequate, or it would have been delivered late (or both).

To help avoid the problem firstly, an effective project plan needs to be developed; and secondly, it needs to be adaptable to change. The first issue is addressed in this Case Study while the second is addressed in the following Case Study.

Overall it is recommended that to minimise the problems and maximise the benefits of the micro-agreement, support should be provided in the contract for an effective project planning activity.

In order to minimise the impact of poor planning it is recommended that the contract provide specific support for firstly, the development of appropriate plans; and secondly, the tracking of activities against those plans. This support should include (either directly or by reference to an appropriate standard) a statement of what the initial planning should include, as well as support for regular reviews and updates to that plan.

It is thus suggested that overall aims should be to:\(^{430}\).

   a. develop a plan that appropriately and realistically covers the software activities;
   b. ensure that all affected groups and individuals understand and commit to the plan; and
   c. ensure that estimates and plans are documented for use in tracking the software activities and commitments.

The issue of tracking and oversight of activities is more fully addressed in Case Study 3 below.

In order to achieve the above aims, some or all of the following specific points should be noted and it is suggested that the contract should include support for these aspects (as appropriate) either directly (or preferably) by reference to appropriate standards such as ANSI/IEEE STD 1058.1, *Standard for Software Project Management Plans*, or other standards such as those listed in Appendix A, *Software Standards* to this document. Note that not all the following points will be relevant to every project and they must be assessed on a case by case basis. The activities list below are derived from the Software Engineering Capability Maturity Model\(^{431}\).

- Software planning should be initiated in the early stages of the project.

   Ideally the initial project plan should be produced before a contract is in place to enable a better assessment of overall contract schedules, cost and quality required. In practice this may require a separate contract to produce the initial


versions of project plans, requirements specifications and other relevant
documents.

- A software life-cycle model with predefined stages of manageable size should be
  identified or defined.

The present case study describes a situation where a classic "waterfall" model was
specified and contractually enforceable through a rigid timeframe for document
delivery. Other lifecycle models should be evaluated depending upon the particular
circumstances of each development. Section 2.4 The Software Lifecycle provides
a general descriptions of some other models which may be applicable.
Publications are available in the literature which provide an analysis of the different
models available and provide guidelines for their selection.\(^{432}\) Standards such as
ISO/IEC 12207 Information Technology - Software Life-cycle Processes, 1995-08-
01 recognise the need for iterative or evolutionary process models and provide
some support for them. Once an appropriate model is identified which gives a
balance between the need for certainty in the contract, and the need for coping
with change throughout the development work, contract clauses can be designed
appropriately.

In order to minimise the risks and maximise the benefits therefore it is
recommended that an appropriate lifecycle model be selected and used and
contractual support be provided for it as appropriate.

- The software development plan should be developed according to a documented
  procedure or set of guidelines.

The purpose of this activity is to maximise the possibility that prior knowledge is
applied to the estimating process. A number of relevant standards and guidelines
are listed in Appendix A. Software Standards and these (and others) should be
evaluated in the context of the particular software development or integration task
being undertaken. One or more of these standards can then be explicitly
referenced in the main contract to provide contractual weight to this activity. As
any schedule or cost estimates are simply that - estimates, it is important that
some rational estimation processes are used in their derivation.

- The software development plan needs to address relevant issues.

In terms of the content of the software development plan it should identify:
developmental processes and standards which are intended to be used during the
software development and integration activities as well as the standards applicable
to the management function; software products to be developed; estimates of
software size; resource estimates including personnel, cost, and development
environment; an estimate of schedules including milestones, and risks associated
with technical, cost, resource, and schedule issues.\(^{433}\)

It should be noted that although the software development plans may help to identify the
best development processes and standards to apply in a particular development, the
parties need to evaluate whether, they should be they are incorporated explicitly into a

166-167

\(^{433}\) this list was extracted from: Weber, V., Paulk, M., Wise, C. & Withey, J. "Key Practices of the Capability
Maturity Model", Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15212,
contract for software development of systems integration work. On the one hand, if they are incorporated there is legally contractual weight given to them rather than just an activity conducted internally within the developers' organisation. If they are incorporated into the contract care needs to be made to ensure that they are not unnecessarily restrictive.

In summary, software development and integration work should involve extensive planning as any deficiencies in the planning process are likely to effect the overall cost, schedule, and quality of the product. Enforcing project planning “best practice” by including appropriate clauses in a contract tied to (for example) appropriate standards may be worthwhile to help prevent problems such as the one identified in this case study.

4.6.5 Legal Analysis of Problems

The micro-agreement in this case consisted of an informal agreement between the Systems Integrator and the Developer to build a prototype of the model in accordance with draft versions of the System Requirements Specification and the System Design Specification. The contract specified that the development was to be undertaken in accordance with a standard “waterfall” model which, in effect meant that the pre-coding\textsuperscript{434} activities such as requirements analysis and design must be completed before the software was coded. The aim of this approach was to ensure that the critical issues of what the system had to do (as defined in the System Requirements Specification documentation), and how it would do it (as defined in the System Design Specification documentation) would be fully investigated before the software coding was begun.

Thus the micro-agreement made aimed to modify the contractual “lifecycle” process as defined in the main contract and build a working software product before the contracted requirements and design documentation were complete. The micro-agreement thus was integral to the planning activity but went against the contracted lifecycle process.

Overall, the parties to this development mutually agreed to vary the main contract by modifying the lifecycle model specified. It thus appears that no legal action or remedy would be readily available, particularly as variations were implemented by mutual consent.

As a consequence of the micro-agreement, unplanned updates to documentation were required and these were produced by the Developer at its cost.

4.6.5.1 Quazi-contract, Unjust Enrichment and Quantum Meruit

Although both parties mutually agreed that documentation could be delivered in draft form pending implementation of a prototype, the additional work conducted by the Developer to update the documentation could possibly be the subject of a quantum meruit claim. Aspects of this claim are discussed in the following paragraphs.

<table>
<thead>
<tr>
<th>Action</th>
<th>Plaintiff</th>
<th>Defendant</th>
<th>Claim</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unjust Enrichment</td>
<td>Developer</td>
<td>Systems Integrator</td>
<td>Claim is that the Systems Integrator was unjustly enriched by the actions of the Developer performing additional work to update the documentation. This would lead to a claim in quantum meruit as a remedy</td>
</tr>
</tbody>
</table>

\textsuperscript{434} “Coding” is the term used to describe the activity of writing a computer software program in a computer "language". An example of a coded ‘statement’ in a language such as "Pascal" is:\footnote{If its_raining_outside then print ("Take your umbrella");}
A claim in quantum meruit is a restitutionary remedy in its own right. It involves "the payment of an amount which constitutes, in all relevant circumstances, fair and just compensation for the benefit or 'enrichment' actually or constructively accepted." Ordinarily, that will correspond to the fair value of the benefit provided (for example, remuneration calculated at a reasonable rate for work actually done or the fair market value of materials supplied). The basis of the claim is founded in the doctrine of unjust enrichment and as such "the right to recover does not depend upon the existence of an implied contract but on a claim to restitution based upon unjust enrichment." Thus, it is not necessary to show that a contract existed, but rather that one party was unjustly enriched at the expense of the other, provided the work done was not of the kind that is normally undertaken gratuitously.

In line with the present case study, it is submitted that recovery, on a quantum meruit basis, may be available to the Developer for its additional work performed in updating the documentation. Specifically, a claim may be available "where one party does work for the benefit of itself or another on the assumption that a contract would ultimately be made between them and where the second party unilaterally abandons the project."

Despite the above Deane J. placed limits on the amounts that can be claimed when he stated that the remuneration must "satisfy the requirements of good conscious and justice which inspire the concept or principle of unjust enrichment." Overall, despite the availability of remedies such as discussed above in this case study, clearly a better solution would have been to prevent the problem from occurring in the first place.

4.6.6 Industry Code of Conduct

A discussion of a proposed industry code of conduct: Information Technology Systems Integration and Outsourcing Code of Conduct is provided in Section 4.11. The discussion provided in that Section addresses issues and makes recommendations for the disclosure of software project planning practices in the information technology systems integration and outsourcing industries.

4.6.7 Findings

In relation to this Case Study key findings are summarised below.

Finding 10: A micro-agreement existed in this case study. (Specifically, the micro-agreement made was explicitly modify the contractual "lifecycle" process as defined in the main contract and build a working software prototype before the contracted requirements and design documentation were complete).

---

436 see Craven-Ellis v Canons Ltd (1936) 2 KB 403 - another quantum meruit case expressly based upon unjust enrichment.
437 Pavey and Mathews Pty Ltd v Paul (1987) 162 CLR 221 at 263-264
439 Sabeco Pty Ltd v North Sydney Municipal Council [1977] 2 NSWLR 880; British Steel Corporation v Cleveland Bridge and Engineering Company Ltd [1984] 1 All ER 504
440 Pavey and Mathews Pty Ltd v Paul (1987) 162 CLR 221
The micro-agreement consisted of an agreement to explicitly modify the contractual "lifecycle" process as defined in the main contract and build a working software product before the contracted requirements and design documentation were complete. The micro-agreement thus was integral to the planning activity but went against the contracted lifecycle process.

Finding 11: The micro-agreement had a significant impact on the development. (Specifically, the product was delivered late).

The impact of the micro-agreement in this case study was (among other things) that the product was tested and delivered late.

Finding 12: The micro-agreement needed to be carefully managed to minimise risk and maximise benefits.

Recommendations on how to minimise the risk and maximise the benefits are provided in the following Section.

4.6.8 Recommendations
In summary, this case study has identified a number of issues, and in order to prevent or minimise the problems and maximise the benefits associated with the micro-agreement, the following recommendations are made:

Recommendation 9: Link the contract to Software Engineering "best practice".

Ensuring that a development is undertaken using software engineering best practice is important. What exactly "best practice" is depends on the particular development and the technologies used. The following recommendation gives an example of one such best practice which should be undertaken in a formal sense in any significant systems integration or outsourcing project.

Recommendation 10: Provide support for effective project planning in the contract.

Due to the complexity of larger information technology systems integration contracts it is essential that effective planning be undertaken so that appropriate resources can be allocated to the tasks.

Recommendation 11: Ensure that an appropriate software development life-cycle model is used.

The particular lifecycle model used can have a significant effect on the outcome of a project. An appropriate lifecycle model should be used for the development depending upon the particular characteristics of the project.
Recommendation 12: A voluntary 'Industry Code of Conduct' should be developed in accordance with the Trade Practices Act, 1974 Part I VB to ensure a Systems Integrator or Outsourcing organisation discloses to the customer the Project Planning activities to be undertaken.

The key concept here is disclosure. If a full and frank disclosure of the project planning activities to be undertaken by the systems integrator or outsourcer is made, then all parties should be in a better position to decide on the terms and scope of any subsequent contract.

4.6.9 Conclusion
This case study has shown how inadequate planning can affect the outcome of a software development activities. It is generally acknowledged that this activity should be undertaken and some contractual support for it may be appropriate.
4.7 CASE STUDY 3 - MICRO-AGREEMENTS IN THE CONTEXT OF SOFTWARE PROJECT TRACKING AND OVERSIGHT

4.7.1 Introduction and Explanation of the Activity

The activity of tracking and overseeing a software development is related to previous case study on project planning. By tracking the software development and integration work, deviations to the plan can be identified and appropriate corrective action put in place early enough to minimise or prevent adverse effects. The activity of Software Project Tracking and Oversight has been summarised as follows:

"Software Project Tracking and Oversight involves tracking and reviewing the software accomplishments and results against documented estimates, commitments, and plans, and adjusting these based on actual accomplishments and results. A documented plan for the software effort is used as the basis for tracking the software activities, communicating status, and revising plans. The software activities are monitored by the software managers on a regular basis. Regular technical reviews and reviews with the project manager and senior management are conducted to ensure that management and staff are aware of the software status and plans, and that issues receive appropriate attention."

The purpose of tracking progress is thus to identify deviations from the plan early enough to enable corrective action to be taken as necessary. The natural outcome of this activity includes the identification of changes necessary to keep the contract on track. Contracts ideally should have a mechanism available to firstly, ensure effective tracking of progress is performed; and secondly, allow changes to be appropriately taken into account.

4.7.2 Example of a Micro-agreement

A contract for a major financial system involved a number of companies of which each was required to produce a substantial software component for integration into the final system. The prime contract between the Buyer and the Systems Integrator included major delivery milestones including the delivery of the final system for formal Buyer testing on a specified date. The time span of the total project was approximately 3 years.

After a significant period from contract start it was becoming obvious to all parties involved that the system could not be delivered with full functionality within the remaining 12 month contract timeframe. In order to minimise the impact, the developer of one sub-system, Developer A, requested that the Buyer prioritise a list of approximately two hundred requirements so that the company could direct its resources to higher priority activities.

As a joint exercise, representatives from Developer A and the Buyer formed a micro-agreement to the extent that both

---

Figure 18: Case Study 3 - Contractual Relationships

---

parties agreed that approximately 25% of the requirements could be classified as "non-essential," (although still required).

Developer A delivered its sub-system on time but with most "non-essential" functionality incomplete or unusable. Developer A claimed that the Buyer, by agreeing to prioritise some individual requirements as "non-essential", was also in fact agreeing to a reduction in the scope of the contracted requirements.

Subsequently it was realised by the Buyer that, although a number of requirements individually were non-essential, collectively they constituted a significant part of the system functionality.

Due to its own commercial pressures, the Buyer reluctantly agreed to enter into a separate contract with the Systems Integrator to fix the problems which were previously classified as "non-essential".

4.7.3 Potential Problems arising from the Micro-agreement
The main problem resulting from the above situation was that the product, although useable to some extent, was difficult for the users to operate under normal operational conditions because a number of the features classified as "non-essential" individually were in fact designed to provide a better user interface collectively. Additional work needed to be performed under a separate contract at additional cost to the Buyer in order to rectify the situation.

4.7.4 Avoiding the Problem
One way of potentially avoiding or minimising the impact of the problem would have been to put in place effective progress tracking and oversight activities so that an "early warning" of potential problems could be given. In general, the earlier that problems are identified the more chance there is of applying corrective action to rectify the problem.

Thus it is recommended that support be provided for tracking and oversight activities in the contract.

In order to ensure that an effective "early warning" system is in place, the following activities have been recommended. It is suggested that providing a contractual base for these activities is likely to ensure that software best practice is accommodated and legally enforceable. It is thus recommended that a large software development or integration contract should provide support for these activities (where appropriate) either directly (or preferably) by reference to appropriate standards. The activities below are derived from the Software Engineering Capability Maturity Model.

Software Development Plans
- A software development plan should be used for tracking the software activities.

Software Development Plans were discussed in the preceding Case Study and accordingly that Section provides a more detailed analysis of issues associated with such planning. The point is however that if a tracking activity is undertaken, expected and actual data is required so that deviations can be identified.

Reviews


443 Ibid
• Formal reviews should be conducted at selected project milestones and at the
beginning and completion of selected stages.

As an example of where a software standard provides support for this activity can
be found in: ANSI/IEEE STD 1058.1, Standard for Software Project Management
Plans. 444 It specifies that “Joint Management Reviews” should to be conducted.
These reviews aim to:

   “a. keep management informed about project status, directions being taken,
technical agreements reached, and the overall status of evolving software
products.

   b. resolve issues that could not be resolved at joint technical reviews.

   c. arrive at agreed-upon mitigation strategies for near and long term risks that
could not be resolved at joint technical reviews.

   d. identify and resolve management level issues and risks not raised at joint
technical reviews.

   e. obtain commitments and acquirer approvals needed for timely accomplishment
of the project.”

The incorporation of periodic formal reviews into the contract provides one way of
identifying potential problems early in the development or integration cycle.

• The software engineering staff and managers should conduct regular reviews to track
technical progress, plans, performance, and issues against the software development
plan.

As an example of where a software standard provides support for this activity,
again can be found in ANSI/IEEE STD 1058.1, Standard for Software Project
Management Plans. 445 It specifies that “Joint Technical Reviews” should to be
conducted periodically. The relevant part of that standard states that:

   “The developer shall plan and take part in joint technical reviews at locations and
dates proposed by the developer and approved by the acquirer. These reviews
shall be attended by persons with technical knowledge of the software products to
be reviewed. The reviews shall focus on in-process and final software products,
rather than materials generated especially for the review.”

Under appropriate circumstances it is thus suggested that software development
contracts should support technical reviews of software products. This can be done
either explicitly in the contract, or by reference to the developers software
development process.

**Items to be Tracked**

From a software engineering point of view, in order to effectively identify problems early in
the development process, at least the following items should be tracked446:

---

444 in its military equivalent MIL-STD-498, Software Development and Documentation, 5 December 1994, at
26.
26.
Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15212, Document CMU/SEI-91-

Paul P. Pamell

128
- The software schedule.
- Software engineering technical activities.
- The software technical, cost, resource, and schedule.
- The software size.
- The software cost.
- The critical target computer resources.

Corrective Action

If an issue arises where the actual data measured from the above items is not consistent with the expected (or planned) data, corrective action should be taken as appropriate to the circumstances.

An example of contract clauses which attempt to provide contractual weight to corrective action procedures can be found in the Government Information Technology Contracts (GITC) \(^447\) as follows:

"The Contractor shall reference and document procedures for corrective action to the Developed Software and associated Documentation including:

(a) adoption of a system to report problems and deficiencies;
(b) examination of problem and deficiency reports to determine their causes, and to prepare corrective measures;
(c) analysis of deficiency trends, to prevent the development of software not conforming to the Developed Software Specifications;
(d) review of corrective measures, to determine their effectiveness;
(e) provision for defining all necessary repetition of testing and reviews to determine the effectiveness of modifications to any item of the Developed Software; and
(f) provision for ensuring that timely corrective action is taken by reviewing deficiencies and tracking their clearance”.

Although the above clauses provide some support for corrective action, it is also important to focus on identifying the problems in the first place. Until problems are identified, they cannot be “corrected”.

Studies have shown that the later a problem is identified in the software development lifecycle, the more difficult and costly it is to fix. The following chart gives an indication of the relative cost to fix problems depending upon which phase of the development lifecycle they were detected in.\(^448\)
Figure 19: Cost of Change V Software Life-cycle Phase

As can be seen from the chart above, for example, if a problem was found during the Requirements phase it would cost approximately 200 times less to fix than if the same problem had been found after the software was put into operation. In summary therefore it is recommended that information technology contracts clearly establish a way for issues to be identified early in the project. Clear procedures also need to be in place to ensure that appropriate corrective action is taken. By specifying software engineering "best practice" in the contract, legally enforceable remedies may be more readily available and may provide an effective way to reduce overall cost and schedule.

From a practical point of view, having contractual support for such activities sends a clear message to all parties that tracking and oversight activities are important and in addition, are likely also to be a powerful risk mitigation tool which allows early insight into potential problems.

4.7.5 Legal Analysis of Problems

What was the Micro-agreement?

In this situation, the micro-agreement consisted of an agreement between the Buyer and Developer A, (who were not in a direct contractual relationship) to classify a number of requirements individually as "non-essential".

Due to an anticipated late delivery date of the product from Developer A which only became apparent towards the later stages of the contract, the Buyer, due to business pressures which required the system to be available, agreed to the re-classification of some requirements as "non-essential".

What was the effect of the micro-agreement?

Overall, the effects of this micro-agreement were that:

- The "non-critical" requirements were not implemented in accordance with the prime contract between the Buyer and the Systems Integrator.
• Because of the micro-agreement, Developer A refused to complete the "non-essential" parts of the system without another explicit contract, and additional payment.
• Due to business pressures, the Buyer provided additional payment to Developer A (via a new sub-contract with the Systems Integrator) to implement the "non-essential" functionality.
• As a result of the micro-agreement, the Systems Integrator (who was not a party to the micro-agreement) was potentially in breach of its prime contract with the Buyer because it could not deliver a system with the full functionality originally specified in that prime contract.
• The main reason that the micro-agreement was put in place was to mitigate the significant risk of not being able to deliver a system at all within the desired timeframe. The agreement was effectively to identify areas of essential functionality needed to provide a basic capability suitable for the Buyer's business needs.
• If sufficient project tracking and oversight activities were implemented then to some extent the micro-agreement may not have been required because the problems should have been identified earlier in the development and hence provided an earlier opportunity for corrective action to occur.

What legal actions may be available to the parties?
A summary of claims and counter-claims is the following table. The merits and chance of success of each of these is discussed in following paragraphs.

<table>
<thead>
<tr>
<th>Action</th>
<th>Plaintiff</th>
<th>Defendant</th>
<th>Claim</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breach of Contract</td>
<td>Buyer</td>
<td>Systems Integrator</td>
<td>Claim is that the Systems Integrator breached its contract with the Buyer by not providing a system complete with all &quot;non-essential&quot; functionality fully implemented.</td>
</tr>
<tr>
<td>Breach of Contract</td>
<td>Systems Integrator</td>
<td>Developer A</td>
<td>Claim is that Developer A breached its contract with the Systems Integrator by not providing a system complete with all &quot;non-essential&quot; functionality fully implemented.</td>
</tr>
<tr>
<td>Estoppel</td>
<td>Developer A</td>
<td>Systems Integrator</td>
<td>Claim is that the Systems Integrator should be estopped from denying that it agreed to classify a number of requirements as &quot;non-essential&quot;</td>
</tr>
<tr>
<td>Claim in quantum Meruit for money had and received</td>
<td>Buyer</td>
<td>Systems Integrator</td>
<td>Claim is that the Systems Integrator was paid twice for the &quot;non-essential&quot; functionality - once via the prime contract, and again through the supplementary contract to implement that functionality.</td>
</tr>
<tr>
<td>Constructive Trust</td>
<td>Buyer</td>
<td>Systems Integrator, Developer A</td>
<td>Claim is that the Systems Integrator and Developer A should be decreed constructive trustees of an additional payment made by the Buyer to the Systems Integrator for implementation of the &quot;non-essential&quot; requirements</td>
</tr>
<tr>
<td>Unconscionable conduct under the Trade Practices Act, 1974</td>
<td>Buyer</td>
<td>Systems Integrator, Developer A</td>
<td>Claim is that it was unconscionable for Developer A to not implement the &quot;non-essential&quot; requirements while at the same time &quot;forcing&quot; the buyer to agree to an additional contract for implementation of those requirements.</td>
</tr>
</tbody>
</table>

Figure 20: Case Study 3 - Summary of Claims
Breach of Contract and Estoppel
In this example the Buyer contracted with the Systems Integrator for the development of a system, the requirements of which were clearly outlined in a requirements specification document. Prima facie, as the Systems Integrator did not deliver the system as specified, the Buyer could claim breach of contract. In its defence however, the Systems Integrator may claim that as a result of the micro-agreement, the Buyer should be estopped from denying that it agreed with Developer A that certain functionality should be classified as 'non-essential' and was thus not required to be implemented.

On the facts, Developer A (and probably by implication, the Systems Integrator) claimed that the Buyer effectively agreed that the "non-essential" functionality need not be implemented under the main contract. From the Buyer's point of view however, the micro-agreement was more likely to be on the basis that the Buyer, because of business pressures to have the system delivered to it and operational, agreed that the system could be delivered without the "non-essential" functionality, provided that that functionality was delivered at a later time although this may not have been explicitly stated. Further facts would be required to establish which of the above two viewpoints is correct.

If it is accepted for the present time that the Buyers view is correct (that is, it agreed to delay but not cancel the delivery of the "non-essential" functionality), it is submitted that the Buyer would be estopped from denying the existence of such a micro-agreement. The issue of consideration is irrelevant under the doctrine of estoppel and thus the practical effect is likely to be that the Systems Integrator and Developer A would be required to implement the "non-essential" functionality, albeit at a later stage.

It is submitted therefore that a claim for breach of contract would not be successful, or more precisely, the Buyer would be estopped from denying that it agreed to extend the time to implement the "non-essential" functionality, and as the time for performance of the obligation had not yet arrived no breach of contract could be claimed. Further discussion on the time for performance issue is provided later in this case study.

Recovery of Money Paid for the "Non-Essential" Requirements
On the facts, the Buyer, for business reasons, made additional payments to Developer A (via a sub-contracted arrangement with the Systems Integrator) after the initial delivery of the system. The reason for this payment was that Developer A refused to implement the "non-essential" functionality because the Buyer implied that it was not required. As the Buyer subsequently discovered that it did in fact need the "non-essential" functionality to effectively operate its business, it agreed to a separate contract to implement those requirements. Can the Buyer recover the additional payment made on the basis that it effectively paid twice for the same functionality?

Possible courses open to the Buyer to recover the additional payment are investigated below. Some of these claims include actions under:

- the Trade Practices Act, 1974
- money had and received
- constructive trust
- economic duress

These and other issues are discussed below.

Trade Practices Act 1974
Could a claim be made under the unconscionability provisions of the Trade Practice Act, 1974? As an example, the Buyer may claim that it was unconscionable for Developer A to
refuse to implement the “non-essential” requirements while at the same time “forcing” the buyer to agree to an additional contract for implementation of those requirements. The *Trade Practices Act, 1974* Section 51AC(1) states that:

(1) A corporation must not, in trade or commerce, in connection with:
(a) the supply or possible supply of goods or services to a person (other than a listed public company); or
(b) the acquisition or possible acquisition of goods or services from a person (other than a listed public company);
engage in conduct that is, in all the circumstances, unconscionable.

Towards the end of the contracted development timeframe it is likely that the Buyer had significant business pressures which required it to put the new software into operation as soon as possible. The Buyer may attempt to claim that the Developer unfairly took advantage of this situation by (1) notifying the Buyer that it could not produce all required functionality in the contracted timeframe, (2) requesting that “non-essential” functionality be identified, and (3) then refusing to implement the “non-essential” functions on the basis that the Buyer had impliedly agreed to this action by allowing the system to be delivered and put into operation without the “non-essential” functionality installed.

Despite this assertion, it is submitted that a claim for unconscionability under the *Trade Practices Act, 1974* Section 51AC(1) would be difficult to achieve because the Buyer, despite being under a degree of business pressure to install the new system, could have insisted on its existing rights under the contract. If the Developer then did not deliver as contracted, presumably contractual remedies would be available.

**Money Had and Received**

Stated simply, it appears that the Buyer overpaid the Systems Integrator for work which was not conducted as specified under the prime contract.

The Buyer may claim that the Systems Integrator and Developer A were thus unjustly enriched by being paid twice for the “non-essential” functionality - once via the prime contract (even thought the non-essential functionality was not delivered under the contract), and again through the supplementary contract to implement that functionality. Is a claim for *money had and received* available in these circumstances?

The action of *money had and received* is available when one person pays money to another under a mistake of fact, when that money would not have been paid if the mistake had been known at the time\(^449\). This action is based on the doctrine of unjust enrichment\(^450\) and in a joint judgement the High Court held that:

> “Before that prima facie liability [to make restitution] will be displaced, there must be circumstances (for example, that the payment was made for good consideration such as the discharge of an existing debt or, arguably, that there has been some adverse change of position by the recipient in good faith and in reliance on the payment) which the law recognises would make an order for restitution unjust”\(^451\).

Overall, it is submitted however that the Buyer would probably not be able to claim under an action for money had and received because it appears that the money was not paid...


\(^{450}\) *Australia and New Zealand Banking Group Ltd v Westpac Banking Corp.* (1988) 62 ALJR 292, per Mason C.J., Wilson, Deane, Toohey and Gaudron JJ.

\(^{451}\) *Australia and New Zealand Banking Group Ltd v Westpac Banking Corp.* (1988) 62 ALJR 292, per Mason C.J., Wilson, Deane, Toohey and Gaudron JJ.
under a "mistake of fact", but rather it was paid as a business decision with full knowledge of the facts at that time.

Constructive Trusts
Could it be that the Systems Integrator and Developer A became trustees of the additional payment made by the Buyer for the "nonessential" requirements to be delivered?

Where there is no explicit trust in place then under appropriate circumstances equity may intervene and create a constructive trust. "A constructive trust is a trust imposed by operation of law, regardless of the intentions of the parties concerned, whenever equity considers it unconscionable for the party holding title to the property in question to deny the interest claimed by another." 452

Despite this, it was explicitly stated by Deane J. in Muschinski v Dodds that a constructive trust is not available under Australian law for the purpose of providing a remedy for unjust enrichment. He said:

"In the United States of America, a general doctrine of unjust enrichment has long been recognised as providing an acceptable basis in principle for the imposition of a constructive trust (see, e.g., Scott, op. cit., vol.V, par.461). It may well be that the development of the law of this country on a case by case basis will eventually lead to the identification of some overall concept of unjust enrichment as an established principle constituting the basis of decision of past and future cases. Whatever may be the position in relation to the law of other common law countries (cf., as to Canada, Pettkus v. Becker (1980) 117 DLR (3d) 257 and, as to New Zealand, Hayward v. Giordani (1983) NZLR 140, at p 148) however, no such general principle is as yet established, as a basis of decision as distinct from an informative generic label for purposes of classification, in Australian law.453

In applying constructive trusts, his Honour also stated that limits should be applied. He said that the constructive trust cannot be used as "a medium for the indulgence of idiosyncratic notions of fairness and justice"; and also that it can be used only when "warranted by established equitable principles or by legitimate processes of legal reasoning, by analogy, induction and deduction, from the starting point of a proper understanding of the conceptual foundation of such principles.454

Despite this, Deane was prepared to find that a constructive trust did exist in the Hospital Products v USSC455 case despite the fact that no fiduciary duty was present. Specifically, in that case Deane J. stated (obiter):

"... USSC was entitled to a declaration that HPI was liable to account as a constructive trustee for the profits of that Australian business in accordance with the principles under which a constructive trust may be imposed as the appropriate form of equitable relief in circumstances where a person could not in good conscience retain for himself a benefit, or the proceeds of a benefit, which he has appropriated to himself in breach of his contractual or other legal or equitable obligations to another."

It is important to note that the Hospital Products case involved a calculated breach of contract. Even though on the face of it Deane J.'s statements in Muschinski and Hospital Products appear to be somewhat contradictory (because in Muschinski he was not prepared to find a constructive trust as a remedy against unjust enrichment, whereas in Hospital Products he was (obiter) prepared to find a constructive trust and to enforce an account of profits), it is submitted that these viewpoints can be reconciled by assuming

452 Evans, M. Outline of Equity and Trusts, Butterworths, 1998, page 239
453 Muschinski v Dodds (1986) 160 CLR 583, Deane J.
454 Muschinski v Dodds (1986) 160 CLR 583, Deane J. at 615
455 Hospital Products v United States Surgical Corporation (1984) 156 CLR 41
that the constructive trust associated with the *Hospital Products* case is an example of a "new or developing category of case" as discussed by Deane J., the calculated breach of contract being the link.

In summary, it appears that "a decree of constructive trust can be made where the defendant has been enriched at the plaintiff's expense and where, in the circumstances, it would be unjust for him to retain that benefit".

Back to the present case study, it is submitted that, if the Buyer could show that there was a calculated breach of contract by Developer A in not implementing the "non-essential" requirements as specified, then a decree of constructive trust may be available.

In practical terms if such an action was to be sustained it would be necessary to look at the terms of the contract to determine the "cost" of the non-essential requirements. It may in fact be quite difficult to separate the cost of these requirements from the remaining functionality at the time the prime contract came into force, especially if the cost of implementing each requirement was not itemised. This would probably be the case in a fixed price contract.

**Economic Duress**

For completeness, the issue of economic duress is also addressed. The Buyer may claim that it was in an untenable position in that, due to the refusal of Developer A to implement the "non-essential" functionality under its main contract (because the Buyer implied that the functionality was not required), it was "forced" to enter into a separate contract for the "non-essential" requirements to be implemented.

The basis of a claim in economic duress has been discussed above in Chapter 3 however it is important in this case to understand that where the party seeking relief has affirmed or acquiesced the change to the contract, a Court will not avoid a contract on the grounds of economic duress, even where it is obvious that the coercion is present.

It is submitted therefore that the actions of the Buyer effectively affirmed the contract and thus would make an action under economic duress unsustainable.

**4.7.6 Suggested Improvements in the Law**

Overall it has been seen from the above discussion that under the facts of this case study the Buyer may have difficulty in claiming back the additional payment made for the "non-essential" requirements to be implemented. Even though it appears that the Buyer made the payment in full knowledge of the facts, it seems that it's hands were some tied because it needed the product for its business and could not afford the time and cost of litigation at that time.

The above discussion has shown that the device of a constructive trust may be available under the circumstances although it is submitted that this result certainly cannot be guaranteed, particularly in light of Deane J.'s comments in *Muschinski v Dodds* (referred to earlier in relation to constructive trusts). Accordingly, it appears that the Buyer in this case may have no suitable remedy at law. Regardless however it shows that vigilance in commercial situations is essential to maintain legal rights.

---

456 *Pavey and Matthews Pty Ltd v Paul* (1987) 162 CLR 221, per Deane J. at 604 when referring to unjust enrichment in relation to new or developing categories.
457 *Evans, M. Outline of Equity and Trusts*, Butterworths, 1998, page 251
4.7.6.1 Time for Performance

In the present case study the Buyer agreed to delay the delivery of a product until a later date. If the actual period associated with that later date was not specified or agreed explicitly, then the concept of a "reasonable time" would apply (as discussed above in Section 3.4.6 above). What then is a reasonable time?

In order to establish a "reasonable time for performance", presumably data must be gathered from the industry, or expert evidence must be called, and that data examined in the context of the specific development. It is submitted that the concept of "reasonable time" may be difficult to establish in the context of large systems integration contracts because firstly, the number of such contracts is relatively few; secondly, published data on the actual time for performance may be difficult and costly to obtain due to the proprietary nature of such information; and thirdly, the time for performance of contractual obligations in a systems integration contract will be somewhat driven by the technology of the day. By the time that the project is complete and time-for-performance data is available, technology has more than likely moved on to such an extent that that data will be somewhat invalid.

Although the concept of "reasonable time" is well known in the law, these problems may make it very difficult and costly to determine in practice. Overall, it is not suggested that another legal concept should be introduced to replace the concept of a "reasonable time" in relation to systems integration contracts, but rather, effort should be made to explicitly agree between the parties what a "reasonable time" is under their particular circumstances.

4.7.7 Industry Code of Conduct

A discussion of a proposed industry code of conduct: Information Technology Systems Integration and Outsourcing Code of Conduct is provided in Section 4.11. The discussion provided in that Section addresses issues and makes recommendations for the disclosure of software project tracking and oversight practices in the information technology systems integration and outsourcing industries.

4.7.8 Findings

In relation to this Case Study key findings are summarised below.

Finding 13: A micro-agreement existed in this case study. (Specifically, two of the parties involved agreed that approximately 25% of the requirements could be classified as "non-essential", although still required).

The micro-agreement consisted of an agreement, between representatives of Developer A and the Buyer, approximately 25% of the requirements could be classified as "non-essential".

Finding 14: The micro-agreement had a significant impact on the development. (Specifically, the product was difficult for the users to operate under normal operational conditions. Additional work needed to be performed under a separate contract at additional cost).

The impact of the micro-agreement in this case study was that the product, although useable to some extent, was difficult for the users to operate under normal operational
conditions because of the number of features not implemented as they were classified as "non-essential". Additional work needed to be performed under a separate contract at additional cost to the Buyer in order to rectify the situation.

Finding 15: The micro-agreement needed to be carefully managed to minimise risk and maximise benefits.

Finding 16: This case study provides an example of where the Buyer of a system, through its actions of affirming the contract (partially because of business pressure), may have no suitable remedy at law.

Finding 17: The legal concept of "reasonable time" may be difficult to establish in the context of large systems integration contracts because firstly, the number of such contracts is relatively few; secondly, published data on the actual time for performance may be difficult and costly to obtain due to the proprietary nature of such information; and thirdly, the time for performance of contractual obligations in a systems integration contract will be somewhat driven by the technology of the day.

Recommendations on how to minimise the risk and maximise the benefits of the micro-agreement in this case study are provided below.

4.7.9 Recommendations

Recommendations associated with this micro-agreement include:

Recommendation 13: Link the contract to Software Engineering "best practice".

Ensuring that a development is undertaken using software engineering best practice is important. What exactly "best practice" is depends on the particular development and the technologies used. The following recommendation gives an example of one such best practice which should be undertaken in a formal sense in any significant systems integration or outsourcing project.

Recommendation 14: Provide support for tracking and oversight activities in the contract.

Tracking the actual progress against the planned progress is an important risk mitigation tool. It helps to identify problems early and hopefully gives sufficient warning to enable corrective action to be taken.
Recommendation 15: This case study provides an example of where the Buyer of a system, through its actions of affirming the contract (partially because of business pressure), may have no suitable remedy at law. Accordingly it is recommended that IT practitioners be made aware that vigilance in commercial transactions is essential to maintain legal rights. In this case study a micro-agreement made between a Buyer and a software developer affected the rights of that Buyer under a separate contract between itself and the Systems Integrator.

Recommendation 16: A voluntary "Industry Code of Conduct" should be developed in accordance with the *Trade Practices Act, 1974* Part IVB to ensure a Systems Integrator or Outsourcing organisation discloses to the customer the tracking and oversight activities to be undertaken.

The key concept here is disclosure. If a full and frank disclosure of the tracking and oversight activities to be undertaken by the systems integrator or outsourcer is made, then all parties should be in a better position to decide on the terms and scope of any subsequent contract.

4.7.10 Conclusion

In conclusion, it is generally acknowledged within the information technology industry that tracking progress can help to lower the risk associated with a project by providing visibility of problems early. As a consequence, it is suggested that to ensure that such activities are performed, ideally contractual support should be provided. In addition it is also suggested that full disclosure of such activities should be the subject of a voluntary Industry Code of Conduct under Section 51AC of the *Trade Practices Act, 1974*. 
4.8 CASE STUDY 4 - MICRO-AGREEMENTS IN THE CONTEXT OF SUBCONTRACTOR MANAGEMENT

4.8.1 Introduction and Explanation of the Activity

In information technology systems integration and outsourcing work, subcontracting portions of the work is important to ensure that sufficient expertise and resources are available to complete the job. Potential problems may arise however from this way of doing business due to many factors ranging from inadequate requirements specifications to unrealistic estimates of cost or schedule. The systems integration or outsourcing role often includes the selection and management of subcontractors and potentially may introduce a high level of risk into a development if not managed appropriately. This activity of managing subcontractors has been described as follows:

"Software Subcontractor Management involves selecting a software subcontractor, establishing commitments with the subcontractor on the work to be performed, coordinating activities with the subcontractor, and tracking and reviewing the subcontractor’s performance and results. The subcontractor is selected on its ability to perform the work. A documented agreement covering the technical and non-technical (e.g. legal, financial and administrative) requirements is established and is the basis for managing the subcontract. Regular technical and management reviews are conducted to ensure that management and staff of both organisations are aware of the software status and plans, and that issues receive appropriate attention." 459

4.8.2 Example of a Micro-agreement

The Systems Integrator for a large project had difficulty in obtaining the final documentation set (including certain performance data) from Subcontractor A. As a consequence, a micro-agreement was formed between the Systems Integrator and Subcontractor B that Subcontractor B could use initial performance estimates from a prototype design from Subcontractor A rather than actual measured data of the final subsystem to be delivered by Subcontractor A.

Subcontractor B, in reliance on the estimates of performance for the sub-system developed by Subcontractor A, undertook development of its sub-system. Unfortunately the performance estimates were found to be incorrect to the extent that data throughput needed to be approximately 25% higher than the estimated throughput provided by the Systems Integrator. The inaccuracy in the data was caused by the fact that the prototype was designed to be run on a higher power development computer whereas the final system was to be run on slower computers with specialised interfaces.

As the software sub-system developed by Subcontractor B was designed to run on a separate computer to that of Subcontractor A, in order to achieve the required data throughput the Systems Integrator, at its cost, upgraded the computer hardware used to run the software developed by Subcontractor B, although no software upgrades were required.

4.8.3 Potential Problems arising from the Micro-agreement

In this case study, documentation to be provided by Subcontractor A was not identified in the contract as critical to the success of the integrated system. The micro-agreement consisted of an agreement between the Systems Integrator and Subcontractor B that Subcontractor B could use preliminary performance estimates in the design of its sub-system.

As a result of the micro-agreement,

1. Subcontractor B used preliminary performance estimates to design its sub-system. This preliminary data estimated the quantity of data per second which needed to be transferred between the two sub-systems when connected together. These preliminary performance estimates were incorrect.

2. The Systems Integrator incurred additional cost in upgrading hardware to run the sub-system developed by Subcontractor B.

3. Although no software changes were involved, additional work needed to be performed by both Subcontractor B and the Systems Integrator to test that sub-system B performed correctly on the upgraded hardware.

4.8.4 Avoiding the Problem

The above is one example of a problem which could arise in a systems integration activity involving the development and integration of subcontracted components. Solid management of the subcontract process is thus usually necessary to achieve a successful, fully integrated product. Activities which are aimed at achieving this management are often defined and put in place within organisations although may not necessarily carry contractual weight.

As a consequence therefore it is recommended that support be provided to aid effective subcontract management.

Specifically, a number of activities identified below should be conducted. The activities are generally recognised in the software engineering community as being associated with the management of subcontracts. Although the activities listed are mainly the concern of the prime contractor (which may be the systems integrator) and to that extent are somewhat unilateral, in some instances support from the subcontractor will be required. Depending upon the circumstances of each particular development, support for the following activities may be desirable either explicitly in the contract, or by reference to appropriate standards, procedures and guidelines. This list is derived from published software engineering data\(^\text{400}\).

Before the “Implementation” Subcontract is Let


Paul P. Parnell 140
Before a major systems integration contract is let, the following activities should be undertaken:

- The work to be subcontracted should be defined and planned according to a documented procedure. This has previously been discussed in Case Studies above however, once again, planning is essential to the success of any large software contract. Appendix A. Software Standards lists some standards which may be useful in this process.
- The software subcontractor should be selected according to a documented procedure, based on a complete evaluation of the subcontract bidder's ability to perform the work. As an example, the Capability Maturity Model\textsuperscript{461} provides an evaluation criteria for undertaking this work and a summary of this model is provided in Appendix C - List of Software Engineering Activities".
- A documented subcontractor's software development plan, which covers (directly or by reference) the appropriate items from the prime contractor's software development plan, should be reviewed and approved by the prime contractor. This document can used for tracking the software activities and communicating status after a contract is in place.

In relation to the above three points, in some cases it may be appropriate for an initial contract to be let with the aim of producing draft requirements specifications, project plans and other relevant documentation. Once this information is available work associated with the various subcontracts may be more easily defined.

In a large developments therefore, it is recommended that consideration be given to having two (or more) contracts - the first phase (preparation contract) being to derive the documentation mentioned above; with the second phase being for implementation of the system based on that documentation.

This approach provides an opportunity for requirements specifications and other related documentation to be more fully defined before an "implementation" contract is put in place. The development is thus likely to be more flexible in terms of ensuring that requirements are identified more fully before the final contract is in place. All parties are potentially in a better position as firstly, the buyer still has the advantage of being able to approach multiple software developers in a competitive environment for the implementation contract; and add or delete requirements as appropriate. The developer also has a more detailed specification upon which to base its quotation for implementing the system, and hence has a better ability to estimate cost and schedule more accurately.

In order to perform the above activities effectively, various standards, procedures and guidelines are available which may help. Appendix A lists some of these and as an example, the following may be useful to help establish a requirements baseline and to develop effective software development and integration plans. In either (or both) a preparation contract or an implementation contract these (or other) standards may be used to define the required level of detail.

\textbullet\ 1062-1993 IEEE Recommended Practice for Software Acquisition (ANSI)
\textbullet\ 1233-1996 IEEE Guide for Developing System Requirements Specifications (ANSI)
\textbullet\ ANSI/IEEE STD 830, Recommended Practice for Software Requirements Specifications
\textbullet\ ANSI/IEEE STD 1058.1, Standard for Software Project Management Plans

Changes to the Contract

- Changes to the subcontracted scope of work, subcontract terms and conditions, and other commitments should be resolved according to a documented commitment review procedure involving affected groups of both the prime contractor and the subcontractor.

As described in chapter one it is important to be able to cope with change. This can be done a number of ways including those discussed above under Case Study 1 - Requirements Management. As discussed in that case study, correct software development lifecycle selection is an important part of any significant development.

Reviews and Audits

As discussed in Chapter 1, reviews and audits can be a technically effective and cost effective way of identifying problems in a system early. To that end the following practices are recommended and ideally should be given contractual weight where appropriate.

- The prime contractor's management should conduct regular status/co-ordination reviews with the subcontractor's management.
- Periodic technical reviews and interchanges should be held with the subcontractor.
- Formal reviews to address the subcontractor's software engineering accomplishments and results should be conducted at selected milestones and at the completion of selected stages.
- The prime contractor's software quality assurance group should monitor the subcontractor's software quality assurance activities according to a documented procedure.
- The prime contractor's software configuration management group should monitor the subcontractor's activities for software configuration management according to a documented procedure.
- The subcontractor's performance should be evaluated on a periodic basis and the evaluation should be reviewed with the subcontractor.

In relation to the above points it is recommended that specific support be provided in the contract to ensure that the above activities are performed effectively. That is, reviews and audits should be explicitly called for under the contract, and relevant data should be specified to and made available by the subcontractor to the prime contractor for assessment.

Case Study 3 Tracking and Oversight gave some examples of particular standards which may be called up within the contract which define regular management and technical reviews to be conducted. A number of other standards may also be applicable in this regard and some of these are listed in Appendix A. Software Standards. As an example, 1028-1988 (R1993) IEEE Standard for Software Reviews and Audits may be useful, depending upon the particular circumstances of a development.

Acceptance

- The prime contractor should conduct acceptance testing\(^{462}\) as part of the delivery of the subcontractor's products according to a documented procedure.

It is important to ensure that an agreed acceptance procedure and acceptance criteria is identified in the contract to avoid disputes over the delivery state of the

\[^{462}\text{that is: "(1) Formal testing conducted to determine whether or not a system satisfies its acceptance criteria and to enable the customer to determine whether or not to accept the system; or (2) Formal testing conducted to enable a user, customer, or other authorised entity to determine whether to accept a system or component."}^{\text{IEEE Standard 610.12-1990 Glossary of Software Engineering Terminology (ANSI).}}\]
system. Any large system is likely to contain defects when delivered, and Chapter 2 provided an indication as to the number of faults likely to be present within a system. In order to cope effectively with this situation it is thus important that any contract clearly define the acceptance criteria. Further discussion of this issue is provided in Case Study 5 below which addresses a specific situation where a micro-agreement changed the acceptance criteria and caused a system to be delivered with potentially a greater number of faults than initially agreed.

From a software engineering point of view a common way of ensuring that there is some degree of objective criteria upon which to accept the system is to undertake a number of activities as follows. Firstly, it is necessary to ensure that requirements are written in a manner which is testable. Various standards listed in Appendix A. Software Standards provide support for writing adequate requirements specifications. As an example, a requirement contained in a software specification which states: “the System shall process 980,000 transactions per second” is testable (assuming ‘transaction’ is adequately defined, and appropriate test tools are available), whereas if the requirement was stated as: “the system shall process an adequate number of transactions per second” is not testable without further information being provided.

Once a testable specification is available, tests can be developed for each of the requirements in that specification. Again, various standards, some of which are listed in Appendix A. Software Standards provide support for testing and verification activities, and these can be called up in the contract explicitly or by reference if required.

Any defects subsequently identified during the testing activities can be documented and classified according to standard definitions, and applied to (for example) an acceptance criteria which allows only a defined number of defects of a particular priority to be present in the system.

Further discussion of this and related issues are provided in the following Case Study - Software Quality Assurance.

In summary, consideration should be given to ensuring that the points mentioned above are addressed and that sufficient support is provided in the contract for relevant activities.

### 4.8.5 Legal Analysis of Problems

Based on the facts provided in this case study, a number of actions may be available. These are summarised in the following table, and discussed further below.

<table>
<thead>
<tr>
<th>Action</th>
<th>Plaintiff</th>
<th>Defendant</th>
<th>Claim</th>
</tr>
</thead>
<tbody>
<tr>
<td>Misleading and Deceptive Conduct - Trade Practices Act, 1974, s.52</td>
<td>Systems Integrator</td>
<td>Subcontract or A</td>
<td>Claim is that Subcontractor A, in providing inaccurate performance estimates, was misleading and deceptive</td>
</tr>
<tr>
<td>Breach of Contract</td>
<td>Systems Integrator</td>
<td>Developer A</td>
<td>Claim is that Developer A breached its contract with the Systems Integrator by not providing a system complete with all “non-essential&quot; functionality fully implemented.</td>
</tr>
</tbody>
</table>
Trade Practices Act

In this case study, performance estimates were required to be delivered by Subcontractor A under its contract with the Systems Integrator however (due to reasons unknown), Subcontractor A was unwilling or unable to deliver the final estimates in an appropriate timeframe for Subcontractor B to make use of them. As a consequence, a set of preliminary estimates was used by Sub-contractor B as the basis of its design. These estimates were provided by Subcontractor A in the early stages of the project before any contract was in place.

The issue is whether it can be claimed that Subcontractor A was guilty of misleading or deceptive conduct within the meaning of Section 52 of the Trade Practices Act, 1974. More precisely, the question is: Did Subcontractor A mislead or deceive the Systems Integrator and Subcontractor B into believing that the preliminary performance estimates were correct, when in fact they were not correct?

In Westsub Discounts v Idaps Australia Limited claims were made against a supplier of a computer system for a number of actions including breach of Section 52 Trade Practices Act, 1974. In that case a contract was put in place between the supplier (Idaps) and the purchaser (Westsub) for the development of a computer system for processing video store hire transactions. During pre-contractual negotiations between the parties, Idaps made certain claims based on a previous system which had been developed for libraries in general.

In summary, Westsub claimed that pre-contractual representations made by Idaps induced it to enter into the contract. Specifically, it was claimed that:

"(a) the application software known as the Technocrat would fulfil the specific requirements of the Applicant’s business; (c) the maximum response time for each terminal would be no more than five (5) seconds; (d) a debtor’s interface would be supplied and implemented with minimal delay; (e) a pricing matrix would be supplied and implemented with minimal delay; (h) the whole computer system would be able to accommodate fifteen terminals with the same 5 second response time guarantee."\(^{464}\)

The technical problems in this case revolved around the inadequate performance of the system. Inaccurate performance estimates were provided by Idaps (the supplier) and relied on by Westsub (the purchaser) and in fact induced Westsub to enter into the contract. Westsub made it clear from the outset that it required certain performance from the system to be developed. Idaps went to lengths to avoid contractual liability by ensuring that such performance figures were not included as part of the contract per se and as such, no contractual liability was found by the Court. Before the contract was entered into however Idaps attempted to minimise the concerns of Westsub by making certain representations about performance - and it was these representations which caused the Court to find in Westsub’s favour.

\(^{463}\) Westsub Discounts Pty Limited v Idaps Australia Limited (1990) 17 IPR 185

\(^{464}\) Westsub Discounts v IDAPS Australia Limited (1990) 17 IPR 185; para 75
In relation to the specific pre-contractual representations, Woodward J. found that a statement made by ldaps to the effect that "we are confident the installation will be a successful one" expressly indicated to Westsub that the system to be developed would meet Westsub's business needs. In addition, before the contract was signed, Westsub sent a letter to ldaps requiring clarification that a 5 second response time would be achieved. ldaps did not respond to Westsub's letter but rather delegated a junior non-technical employee to respond orally. The Court found that the employee did make some attempt to discount the performance figures but overall these attempts amounted to "a muted reservation" which was not "sufficiently clear or emphatic to raise doubts".

A key point is that the failure of ldaps to expressly deny that it could meet performance criteria set out in the Buyer's pre-contractual letter was sufficient to found an action for misleading and deceptive conduct under Section 52 of the Trade Practices Act, 1974. This was despite the fact that ldaps took steps to ensure that such performance figures were not embedded into the eventual contract which was signed by the parties.

In relation to the present case study, the Westsub Discounts v ldaps Australia Limited case is important because in both that case, and the present case study under consideration, estimates of performance provided by one party were relied upon to some degree by another party.

In the Westsub Discounts case the buyer was induced to enter into a contract as a result of performance estimates which were provided by the developer of the system. In the present case study the Systems Integrator and Subcontractor B relied upon preliminary performance estimates provided by Subcontractor A. In order to establish whether Subcontractor A could be subject to an action for breach of Section 52 of the Trade Practices Act, 1974 additional facts would be required to establish to what degree the parties relied upon these preliminary performance figures. In the present case study all parties understood that the performance figures were preliminary and that final figures were to be provided under the contract. On the other hand no attempt was made by Subcontractor A to explicitly deny or discount these preliminary performance figures.

In summary, points to note from the Westsub Discounts case include:

- "an area of concern is that the decision appears to extend strict liability to the supplier"
- broad statements such as ldaps claim that "we are confident the installation will be a successful one" in the past would have been regarded as "mere puffs". This case effectively elevated such statements to the level of misleading and deceptive conduct under Section 52 of the Trade Practices Act, 1974.
- "the Court seems to have thrown upon the vendor an obligation in every case to positively go into the user and find out the user's requirements before offering for sale its own product. This is surely to impose a duty of professional adviser upon those who are nothing more than salespeople who, although required to be honest, do not have a fiduciary duty at law"

Overall, cases such as Westsub Discounts Pty Limited v ldaps Australia Limited (and others such as Madeley Pty. Limited (t/a The Venture Group) v Touche Ross & Co.

---

465 Westsub Discounts v IDAPS Australia Limited (1990) 17 IPR 185
466 These points were raised in Knight, P. and Fitsimons, J: Negotiating Information Technology Contracts, at page 109 being papers presented at a seminar held Sydney 14,15 September 1995.
467 Knight, P. and Fitsimons, J: Negotiating Information Technology Contracts, being papers presented at a seminar held Sydney 14,15 September 1995 at page 109.
468 Westsub Discounts Pty Limited v Idaps Australia Limited (1990) 17 IPR 185
(Unrep))\(^{469}\) have placed a considerable onus on developers (and by extension Systems Integrators and outsourcers) to provide accurate information on the predicted performance of computer systems to be developed. In that light, "mere puffs" such as "we are confident the installation will be a successful one" have been elevated to the position of misleading and deceptive conduct under Section 52 of the Trade Practice Act, 1974.

On balance it is submitted that the actions of Subcontractor A, in providing preliminary performance data to the other parties, could constitute misleading and deceptive conduct under the Trade Practices Act, 1974 if it could be shown on the facts that those parties relied upon the information to their detriment. In its defence however Subcontractor A may claim that all parties understood that the estimates were preliminary only and subject to formal confirmation through a contractually deliverable set of performance data.

If a claim for misleading and deceptive conduct was successful, what remedies would be available? Woodward J., in Westsub Discounts v Idaps Australia Limited\(^{470}\) stated in relation to the question of damages that:

```
140. It is now well established that the appropriate measure of damages for Breach of Part V of the Trade Practices Act is, generally speaking, the same as in the case of deceit. The aim is, so far as possible, to put the applicant back in the position it would have been in if the misleading conduct had not occurred, or had not induced the applicant to act to its detriment.

141. The applicant is therefore entitled to compensation for any reasonably foreseeable losses which it incurred directly by reason of its having entered into the contract with the respondent. In assessing these damages, allowances must be made for any benefit which the applicant received from the contract. The applicant is also entitled to any reasonably foreseeable indirect or consequential damage which it suffered as a result of its entering into the contract; see Gates v The City Mutual Life Assurance Society Ltd (1986) 160 CLR 1 at 12-13. Such damages could include terms such as interest on borrowings or earnings which might have been made from other sources if its capital or borrowings had not been invested in the comparatively unproductive computer system.

142. However, the applicant is not entitled to damages for lost expectations. It is not entitled to damages to compensate it for the profits which it might have made if the representations had turned out to be accurate. Such damages would be appropriate in an action for breach of contract but not in deceit."
```

Despite this it is interesting to note that one of the problems which confronted Westsub was providing evidence of damage to the Court. Lack of evidence in that regard was an important factor because despite the Court finding in its favour, the actual damages awarded to Westsub were not particularly significant taking all the circumstances into account.

Thus it appears that based on the facts of this case study, a claim against Subcontractor A may be possible under Section 52 of the Trade Practices Act, 1974, and if so, damages would be available to, so far as possible, put the Systems Integrator and Subcontractor B back in the position they would have been in if the misleading conduct had not occurred (subject to evidence of loss) being available.

**Estoppel**

Estoppel has already been discussed in detail in other parts of this document however in the context of this case study, the question is whether Subcontractor B can claim that

\(^{469}\) Madeley Pty, Limited (via The Venture Group) v Touche Ross & Co. (Unrep), Sup. Ct. of Vic, McGarvie J, 21 Dec 1989, No. 498 of 1988

\(^{470}\) Westsub Discounts v IDAPS Australia Limited (1990) 17 IPR 185
either or both the Systems Integrator and Subcontractor A should be estopped from denying that the preliminary performance estimates they provided to sub-contractor B were valid.

Subcontractor B relied on the preliminary performance data provided to it and used those figures as the basis for developing its'sub-system. It appears that the final performance figures were due to be provided as a contractual deliverable by Subcontractor A but as they were not delivered in time the preliminary estimates were used instead.

Looking at estoppel itself, as indicated in previous Sections, the basic principle of estoppel is that "the law should not permit an unjust departure by a party from an assumption which he has caused another party to adopt or accept for the purpose of their legal actions. The departure will be unjust if the other person having acted or abstained from acting on the basis of the assumption, would suffer a detriment if the opposite party were afterwards allowed to set up rights against him inconsistent with that assumption." 471

Prima facie a case could possibly be argued on the above grounds, that is, if Subcontractor A and the Systems Integrator caused Subcontractor B to assume that the preliminary performance figures were either unlikely to change significantly, or in fact were to be used as the final figures. The micro-agreement discussed in the facts of this case study was that the Systems Integrator and Subcontractor B came to an agreement that the preliminary performance data could be used. It is unclear on the facts whether there was any indication that these figures would differ significantly to the final data to be delivered.

In support of such a claim of estoppel, based on the Waltons v Maher 472 decision, commentators including Elliott 473 have indicated that: "Practitioners must now be aware that conduct and statements during the course of pre-contractual negotiations can lead to binding obligations and an award of damages, even if the parties have not entered into a contract." 474 Based on this premise it could be argued that the conduct of the Systems Integrator in allowing Subcontractor B to rely on the (incorrect) performance data, could be the basis of a claim in estoppel.

Following on from this point however the total circumstances need to be looked at in determining the question of estoppel. Specifically, factors which need to be addressed include 475:

- whether Subcontractor B in fact acted to its detriment by operating on the assumption that the preliminary performance data supplied was correct;
- whether the relationship of the parties included some form of reliance by Subcontractor B on the other parties;
- whether there was a duty for the Systems Integrator and Subcontractor A to give true and accurate performance data at the relevant time; and

472 Waltons Stores (Interstate) Ltd v Maher & Anor. (1988) 62 ALJR 110
473 see Elliott, Paul: "Developments in Equitable Estoppel since Waltons Stores", being a paper delivered at the BLEC Workshop: "The Law of Contract Disputes, 1994"
474 ibid
475 ibid
whether the Systems Integrator and Subcontractor A allowed Subcontractor B to continue to act on its false assumption after the inaccuracy of the data was known.

If the above points can be satisfied then it is likely that the equitable notion of unconscionability or fairness will apply, and the Systems Integrator and Subcontractor A may be estopped or prevented from going back on the state of affairs he had induced another to accept or act upon, or in other words, prevented from denying that the preliminary performance estimates were accurate.

Despite the above, in order to claim estoppel it is reasonably clear that "there must be some sense of inherent injustice, of clear unfairness existing in the circumstances". In defence of the action in estoppel therefore it may be claimed that all parties knew that the estimates provided were just that - estimates - and that the final data was to be provided as a formal contractual deliverable.

Overall, it is submitted that there may be reasonable grounds for finding an action in estoppel in this case study although additional facts would be needed to fully investigate the chance of success. The primary question however is whether in all the circumstances of the case there was "some sense of inherent injustice, of clear unfairness existing in the circumstances".

Breach of Contract
Based on the facts of this case study, Subcontractor A had an obligation under the contract to formally deliver performance data, but at the time the problems occurred with Subcontractor B had not done so. Was Subcontractor A in breach of contract?

Generally speaking, any contractual obligation has a time for performance associated with it. In other words, a person cannot be in breach of a contractual obligation unless the time for performance of that obligation has passed. On the facts of the case study it is not indicated whether such time has passed and accordingly additional information would be required to answer the question absolutely. Despite this a number of points can be made and these are discussed below.

When the Time for Performance is Express
If the time for performance of the obligation is expressly identified in the contract then two situations can occur. If that time is clearly identified in the contract as being an essential term to the extent that the other party requires absolute compliance and in essence would not have entered into the contract if there was any doubt, then that party has an immediate right to terminate the contract once that expressed time has passed.

If however the term of the contract is not an essential term to the above extent, then once the expressed time for performance has passed, provided reasonable notice has been given and that time has expired, the contract may be terminated. The key point is that even if a contract expressly states a time for performance, the fact that that time has passed does not necessarily give a legal right to terminate the contract. The reason is that although the contract may identify (for example) a calendar date, the legal right to terminate stems not from that date but rather from the time for performance which may be governed by additional requirements of reasonable notice etc. Consequently, if a party purports to terminate a contract for breach of contract, if the time for performance of the other party has not passed then the terminating party itself will be in breach. In practical

476 ibid
477 ibid
478 Foran v Wight (1989) 168 CLR 385
terms therefore usually a right to terminate a contract will crystallise after firstly, the contracted time for performance has expired, secondly a notice to terminate has been issued which allows the other party a reasonable time to perform, and thirdly when that reasonable time has expired.

In the present case study for example, if the time for delivery of the final performance estimates from Subcontractor A was 1 June then provided an appropriate notice to terminate was issued, the contract could be terminated on (1 June + a reasonable time). The question of what constitutes a reasonable time itself has difficulties associated with it and has been discussed elsewhere in this document including Section: 3.4.6 Time for Performance.

When the Time for Performance is Implied
When this is no time for performance expressly called for in the contract then usually a term is implied such as to make it a "reasonable time". Issues associated with an implied time for performance have been previously discussed in Section 3.4.6, and in Case Study 3 at Section 4.7.6.1. In summary however if no time for delivery of the final performance data has been expressly incorporated into the contract then the concept of a reasonable time will apply. The Systems Integrator must in that case ensure that a reasonable time has passed before terminating its contract with Subcontractor A, otherwise it may be in breach itself.

4.8.6 Suggested Improvements in the Law

4.8.6.1 Misleading and Deceptive Conduct
As discussed above, cases such as Westsub Discounts Pty Limited v Idaps Australia Limited\(^7\) and Madeley Pty. Limited (t/a The Venture Group) v Touche Ross & Co. (Unrep)\(^8\) have effectively elevated "mere puffs" to the status of misleading and deceptive conduct under Section 52 of the Trade Practice Act, 1974. Has the law gone too far? Everyone knows that salespeople tend to sell their products in favourable light and play down inadequacies of their products. Fair enough if fraudulent or negligent misstatements are made about a product, but should the law go beyond that to effectively impose a strict liability for claims made if those claims are honest and accurate to the extent possible given circumstances known at the time they were made? It will be interesting to see how the law develops in future in this area and whether this trend continues.

4.8.7 Industry Code of Conduct
A discussion of a proposed industry code of conduct: Information Technology Systems Integration and Outsourcing Code of Conduct is provided in Section 4.11. The discussion provided in that Section addresses issues and makes recommendations for the disclosure of subcontractor management practices in the information technology systems integration and outsourcing industries.

4.8.8 Findings
In relation to this Case Study key findings are summarised below.

\(^7\) Westsub Discounts Pty Limited v Idaps Australia Limited (1990) 17 IPR 185
\(^8\) Madeley Pty. Limited (t/a The Venture Group) v Touche Ross & Co. (Unrep), Sup. Ct. of Vic, McGarvie J, 21 Dec 1989, No. 496 of 1988

Paul P. Parnell 149
Finding 18: A micro-agreement existed in this case study. (Specifically, the micro-agreement consisted of an agreement between the Systems Integrator and Subcontractor B that Subcontractor B could use preliminary performance estimates in the design of its sub-system).

The micro-agreement was formed between the Systems Integrator and Subcontractor B. The agreement was that Subcontractor B could use initial performance estimates from a prototype design from Sub-contractor A rather than actual measured data of the final subsystem to be delivered by Subcontractor A.

Finding 19: The micro-agreement had a significant impact on the development. (Specifically, the Systems Integrator incurred additional cost in upgrading hardware to run the sub-system developed by a subcontractor).

The impact of the micro-agreement in this case study was that firstly the Systems Integrator incurred additional cost in upgrading computer hardware to run the sub-system developed by Subcontractor B, and secondly, additional work needed to be performed to test that the system performed correctly on the upgraded hardware.

Finding 20: Cases examining computer system developments have effectively elevated "mere puffs" to the position of misleading and deceptive conduct under Section 52 of the Trade Practice Act, 1974. As a consequence of the law in this regard, predicted computer system performance estimates should not be provided unless they are accurate and correct, or unless heavily cavedated so as to cause no misunderstanding.

Care should be taken when providing estimates of computer performance, and by analogy, any estimates (including schedule and cost).

Finding 21: The micro-agreement needed to be carefully managed to minimise risk and maximise benefits.

Finding 22: Attempting to exclude performance figures from a contract per se may avoid contractual liability, although if pre-contractual claims are made which are misleading or deceptive then an action may be available through other means such as the Trade Practices Act, 1974.

In addition, a failure by a supplier to expressly deny that it can meet performance criteria required by the Buyer can be sufficient grounds to found an action under Section 52 of the Trade Practices Act, 1974.
Recommendations on how to minimise the risk and maximise the benefits are provided in the following Section.

4.8.9 Recommendations

Recommendations associated with this micro-agreement include:

Recommendation 17: Link the contract to Software Engineering "best practice".

Ensuring that a development is undertaken using software engineering best practice is important. What exactly "best practice" is depends on the particular development and the technologies used. The following recommendation gives an example of one such best practice which should be undertaken in a formal sense in any significant systems integration or outsourcing project.

Recommendation 18: Provide support for subcontract management activities in the prime contract.

In information technology systems integration and outsourcing projects it is normal to subcontract parts of the work. Effective control of the subcontracting process needs to be in place to minimise the risks associated with a particular subcontract.

Recommendation 19: A voluntary "Industry Code of Conduct" should be developed in accordance with the *Trade Practices Act, 1974* Part IVB to ensure a Systems Integrator or Outsourcing organisation *discloses* to the customer the subcontract management activities to be undertaken.

The key concept here is disclosure. If a full and frank disclosure of the subcontractor management activities to be undertaken by the systems integrator or outsourcer is made, then all parties should be in a better position to decide on the terms and scope of any subsequent contract.

Recommendation 20: Consider separate contracts - the first for defining the requirements and other documentation as necessary and the second for implementing the software.

Having an initial contract to produce an initial documentation set has the advantage of providing a breakpoint after that contract is complete where the customer is not locked in to a particular supplier or technological solution. It allows a reassessment of issues before entering into a main contract for implementation of the system.
Recommendation 21: As a strategy for avoiding possible action under the *Trade Practices Act, 1974*, predicted computer system performance estimates should not be provided by computer system developers unless they are accurate and correct, or unless heavily caveated so as to cause absolutely no misunderstanding as to their accuracy or correctness or the basis of their derivation.

Recommendation 22: Performance requirements should be explicitly stated in a contract (possibly by reference to an engineering specification) in order to have a contractually binding effect.

Recommendation 23: As a strategy for avoiding possible claims under the *Trade Practices Act, 1974*, if a buyer requests that certain performance criteria of a system be met, then a supplier should *expressly* deny that it can meet these criteria if there is any doubt.

In some respects the above three recommendations are common sense although nonetheless worth stating.

### 4.8.10 Conclusion

Overall, this case study shows that the effective management of subcontracts is an import part of any large development, and particularly in respect of information technology systems integration and outsourcing work. It is recommended that contractual weight be given to enforce subcontractor management activities, or at least the supplier should disclose the activities and control it has over subcontractors.
4.9 CASE STUDY 5 - MICRO-AGREEMENTS IN THE CONTEXT OF SOFTWARE QUALITY ASSURANCE

4.9.1 Introduction and Explanation of the Activity

The field of Software Quality Assurance has undergone significant changes over the past few years and a number of sets of standards, process models, procedures and guidelines have emerged which cover various aspects of the work. Appendix A, Software Standards lists some examples of documents which may be relevant in this regard. Software quality assurance has been defined as follows:

"Software Quality Assurance involves reviewing and auditing the software products and activities to ensure that they comply with the applicable processes, standards, and procedures, and providing the staff and managers with the results of their reviews and audits. The software quality assurance function is required on all projects. The group performing this function is independent of the software groups and project management. A senior manager who is committed to handling all major software quality assurance issues is identified. Where compliance issues exist, the software quality assurance group works with the appropriate managers, including senior management where required, to resolve the issues." 481

4.9.2 Example of a Micro-agreement

The acceptance criteria 482 for a large system was based on agreed limits of "problem reports". Specifically, a software development standard called up in the contract allowed the system to be delivered with a maximum number of known problems although the severity of each problem was to be assessed based on standard criteria. Under the software development standard used, the final system was deemed "acceptable" if the total number of problems of various priority was below specified limits. The following table summarises the limits allowed for various problems in the delivered system 483.

<table>
<thead>
<tr>
<th>Priority</th>
<th>Problem Classification</th>
<th>Allowable Number</th>
</tr>
</thead>
</table>
| 1        | a. Prevents the accomplishment of an operational or mission essential capability; or  
          | b. Jeopardises safety, security, or other requirement designated "critical" | 0                |
| 2        | a. Adversely affects the accomplishment of an operational or mission essential capability and no work around solution is known; or  
          | b. Adversely affects technical, cost, or schedule risks to the project or to life cycle support of the system; and no work around solution is known. | 0                |
| 3        | a. Adversely affects the accomplishment of an operational mission or essential capability but a work around solution is known; or | 10               |

482 "The criteria that a system or component must satisfy in order to be accepted by the user, customer, or other authorised entity." :IEEE Standard 610.12-1990 Glossary of Software Engineering Terminology (ANSI)
483 The above classifications are from MIL-STD-498, Software Development and Documentation, 5 December 1994, at Appendix C.
### Table

<table>
<thead>
<tr>
<th>Priority</th>
<th>Problem Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>a. Results in user/operator inconvenience or annoyance but does not affect a required operational or mission essential capability; b. Results in inconvenience or annoyance for development or support personnel, but does not prevent the accomplishment of those responsibilities.</td>
</tr>
<tr>
<td>5</td>
<td>Any other problem.</td>
</tr>
</tbody>
</table>

Despite the above definitions which were extracted from the software development standard, a micro-agreement was formed between members of the "Problem Report Review Group". This group consisted of members from both the System Integrator and the Developer of a software component, and had the responsibility of assessing each problem and allocating a priority. A micro-agreement was made between the members of this group to change the "Problem Classification" definitions listed in the above table to the following:

- **Priority 1/2**: An error which prevents the accomplishment of a mission requirement, or which jeopardises equipment or personnel safety, as agreed by the Problem Report review group.
- **Priority 3**: An error which adversely affects the accomplishment of an operational or mission essential function, and for which there is an operational workaround.

![Figure 23: Case Study 5 - Contractual Relationships](image)

The overall effect of the above micro-agreement was that classification and prioritisation of problems was effectively put under the control of the Problem Report Review Group. As acceptance of the final system was based on specified limits of the number of problems of various priority, the Problem Report Review Group, by modifying the definitions associated with problem classification, indirectly gave itself a large degree of discretionary power for accepting or rejecting the system (as it was responsible for deciding in which category problems fitted).

The micro-agreement did not explicitly contradict the contract because, even though the contract called up a software development standard, that standard itself allowed "tailoring" of its clauses based on the circumstances of each development.

### 4.9.3 Potential Problems arising from the Micro-agreement

As a result of the micro-agreement:

---

Paul P. Parnell 154
1. The final system was delivered with additional problems that may have been found unacceptable if the standard definitions of priority were used instead of the new definitions which were the subject of the micro-agreement.

2. Additional work needed to be performed to rectify the problems. A separate contract at additional cost was put in place to do this.

**4.9.4 Avoiding the Problem**

Software Quality Assurance activities are naturally an important part of any large software development although it should be remembered that, by themselves, cannot guarantee success. Despite that, it is important that any software quality assurance activities performed are given an appropriate degree of contractual weight.

Accordingly, it is recommended that support be provided in the contract for Software Quality Assurance activities.

In terms of the specific problems identified as resulting from the above micro-agreement it is considered that they may have been avoided or minimised by performing some or all of the software quality assurance related activities described below. In particular, those activities related to checking compliance with standards etc. may be relevant as the changes to the priority definitions should have been identified as "deviations" and notified to management for assessment and corrective action as appropriate.

The following activities are recognised within the software community as being necessary (or at least desirable) to have an effective software quality assurance capability. These are derived from the Capability Maturity Model\(^ {484} \).

- A Software Quality Assurance plan should be prepared for each software project according to a documented procedure.
- The Software Quality Assurance activities should be performed in accordance with the Software Quality Assurance plan.
- The Software Quality Assurance group should participate in the preparation, review, and approval of the project's software development plan, process specifications, standards and procedures.
- The Software Quality Assurance group should review and audit the software engineering activities to ensure process compliance.
- The Software Quality Assurance group should review representative samples of deliverable and designated non-deliverable software products to ensure compliance with the designated process requirements.
- The Software Quality Assurance group should regularly report the results of its reviews and audits to the software engineering staff and managers.
- Deviations identified in the software engineering activities should be documented and handled according to a documented procedure.
- The Software Quality Assurance group should conduct regular reviews of its activities and findings with the customer's Software Quality Assurance personnel.

In terms of ensuring that appropriate coverage is provided in the contract, Appendix A provides a list of standards and guidelines which may be applicable. It is recommended

---

that these (and other) documents be examined and, where applicable, incorporated into the contract either explicitly or by reference.

Despite the above it is important to understand that the fact that a company is accredited to a particular quality standard (such as AS9001), in itself does not guarantee that all required activities are undertaken. Depending upon the wording of the contract, a company may or may not be contractually bound to apply the relevant quality processes to the particular task.

The important point is that merely calling up standards in a contract is insufficient to guarantee success. In particular, it is important that all parts of the standard are explicitly examined for suitability, and tailored appropriately.

4.9.5 Legal Analysis of Problems

4.9.5.1 Introduction

In this case study, a number of issues arise which relate to a micro-agreement formed between the members of a “Problem Report Review Group” which consisted of representatives from each of the parties that had an interest in the development. One of the main tasks of this group was to examine the detail of each defect found in the developed software (prior to delivery and acceptance) and to allocate a ‘priority’ to it based on standard definitions. The prioritised list of all problems would then be used to determine whether the software was suitable for acceptance.

The contract in this case study called for the development to be undertaken using a software development standard MIL-STD-498 Software Development and Documentation. The micro-agreement consisted of an agreement to modify some of the definitions in MIL-STD-498 which relate to priorities of defects.

The MIL-STD-498 standard itself allows (and in fact recommends) tailoring of clauses to suit the particular circumstances of each software development. Specifically, it states that:

“This standard [...] is meant to be tailored by the acquirer to ensure that only necessary and cost effective requirements are imposed on software development efforts.”

More to the point in relation to this case study, MIL-STD-498 also provides for tailoring of the problem report priority definitions. Specifically, it states that the problem report classification definitions are:

“a mandatory part of the standard, subject to the following conditions: 1) these requirements may be tailored by the acquirer, and 2) the developer may use alternate category and priority schemes if approved by the acquirer.”

Based on these clauses, together with the facts of the case study provided above, a list of possible causes of action is provided in the following table.

<table>
<thead>
<tr>
<th>Action</th>
<th>Plaintiff</th>
<th>Defendant</th>
<th>Claim</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breach of Contract</td>
<td>Systems Integrator</td>
<td>Developer</td>
<td>Claim is that the Developer breached its contract with the Systems Integrator by delivering a subsystem which had a greater number of defects than allowable under limits defined in the contracted development standard.</td>
</tr>
</tbody>
</table>

466 MIL-STD-498, Software Development and Documentation dated 4 December 1994, Appendix C.1

Paul P. Parnell 156
<table>
<thead>
<tr>
<th>Action</th>
<th>Plaintiff</th>
<th>Defendant</th>
<th>Claim</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estoppel</td>
<td>Developer</td>
<td>Systems Integrator</td>
<td>Claim is that the Systems Integrator should be estopped from denying that it approved the modification of the &quot;problem classification&quot; definitions.</td>
</tr>
<tr>
<td>Trade Practices Act, 1974, Section 52</td>
<td>Developer</td>
<td>Systems Integrator</td>
<td>Claim is that the Systems Integrator, by agreeing to the changes in the definitions related to defect priority, engaged in misleading and deceptive conduct by later denying that it gave approval for the changes.</td>
</tr>
</tbody>
</table>

**Figure 24: Case Study 5 - Summary of Claims**

4.9.5.2 Breach of Contract

One possibility is that the System Integrator may claim that the Developer breached its contract by delivering a sub-system that had a greater number of defects than allowable under the standard definitions provided by the development standard.

If the contract stated, for example, that "The software development shall be conducted in accordance with MIL-STD-498", then if no further elaboration was provided the first issue would be whether the MIL-STD-498 clauses related to tailoring would be binding. On the face of it appears that the clauses would be binding due the explicit intention of the parties that the standard be included in its entirety without exception.

As the standard allows changes to be made by the 'acquirer', or with its approval, the next issue then is whether the changes to the priority definitions were made by or approved by the System Integrator. As the Problem Report Review Group consisted of members of each of the parties, including the System Integrator, it is unlikely that (on the facts provided), the System Integrator could claim that changes were not appropriately made, or "approved".

The Systems Integrator was a member of the Problem Report Review Group and presumably had actual or constructive knowledge of the workings of that group and was free to raise objections as it saw fit. Further, the software development standard called for in the contract gave the 'acquirer' (that is, the Systems Integrator in this case study) the sole discretion to approve modifications to priority definitions. Through the clauses mentioned above, the Systems Integrator had a significant contractual advantage in being able to control the priority definitions. The fact that the Problem Report Review Group collectively agreed to change the priority definitions is a reasonably clear indication that approval was give by the Systems Integrator, and thus it appears the changes were made in accordance with the contract.

It is thus considered overall that in this case study it is unlikely a claim for breach of contract could be made by the Systems Integrator on the grounds that the Developer failed to deliver a product in accordance with specified standards. On the facts, the Developer delivered a product in accordance with the revised standards validly approved by the Problem Report Review Group, of which the Systems Integrator was a member.

This case study thus gives an example of the hazards of calling up a standard and not tailoring it appropriately to the contract. It is thus recommended that specifications are not blindly called up in the contract without first understanding the impact of doing so. Recognise that standards are usually written as general documents that may require some level of tailoring to match with the specific project aims.
To avoid the problem in the first place the Systems Integrator should probably have tailored the standard to fit appropriately into the context of the particular development, or alternatively not approved the changes.

4.9.5.3 Estoppel, Trade Practices Act, 1974
A discussion of the operation of both estoppel and the Trade Practices Act, 1974 has previously been provided (for example in Sections 3.2.7, and 3.2.6.2.1 above). In the context of the present case study, it seems reasonably clear that the Systems Integrator would be estopped from denying that it gave approval for the changes to the priority definitions. As a member of the Problem Report Review Group, it would have been a party to the activities of that group and would have been actively involved in the Group's decision making process.

Even if the representative of the System Integrator was silent when the changes were proposed then the Developer may claim that that silence itself may constitute a breach of Section 52 of the Trade Practices Act, 1974 as conduct which was misleading and deceptive or at least conduct which was likely to mislead or deceive.

4.9.6 Industry Code of Conduct
A discussion of a proposed industry code of conduct: Information Technology Systems Integration and Outsourcing Code of Conduct is provided in Section 4.11. The discussion provided in that Section addresses issues and makes recommendations for the disclosure of software quality assurance practices in the information technology systems integration and outsourcing industries.

4.9.7 Findings
In relation to this Case Study key findings are summarised below.

Finding 23: A micro-agreement existed in this case study. (Specifically, A micro-agreement was made between the members of the "Problem Report Review Group" to change the "Problem Classification" definitions originally defined in standards called up in the contract).

A micro-agreement was made between the members of the "Problem Report Review Group" to change the "Problem Classification" definitions defined in a standard called up in the main contract.

Finding 24: The micro-agreement had a significant impact on the development. (Specifically, the final system was delivered with a greater number of defects than necessary. Additional work was needed to rectify these problems. A separate contract at additional cost was put in place to do this)

The impact in this case study was that the final system was delivered with additional problems which may have been found unacceptable if the standard definitions of priority were used instead of the new definitions which were the subject of the micro-agreement; and additional work needed to be performed to rectify problems in the system as a result of the new problem classification definitions.

Paul P. Parnell

158
Finding 25: The micro-agreement needed to be carefully managed to minimise risk and maximise benefits.

Finding 26: Silence itself may constitute a breach of Section 52 of the Trade Practices Act, 1974 as conduct which is misleading or deceptive (or at least conduct which was likely to mislead or deceive).

Recommendations on how to minimise the risk and maximise the benefits are provided in the following Section.

4.9.8 Recommendations

In the context of this case study the following recommendations are made:

Recommendation 24: Link the contract to Software Engineering "best practice".

Ensuring that a development is undertaken using software engineering best practice is important. What exactly "best practice" is depends on the particular development and the technologies used. The following recommendation gives an example of one such best practice which should be undertaken in a formal sense in any significant systems integration or outsourcing project.

Recommendation 25: Prove support in the contract for software quality assurance activities.

Quality Assurance is one of the fundamental practices which is essential to ensuring a product is delivered which is consistent with customer expectations.

Recommendation 26: A voluntary "Industry Code of Conduct" should be developed in accordance with the Trade Practices Act, 1974 Part IVB to ensure a Systems Integrator or Outsourcing organisation discloses to the customer the Quality Assurance activities to be undertaken.

The key concept here is disclosure. If a full and frank disclosure of the quality assurance activities to be undertaken by the systems integrator or outsourcer is made, then all parties should be in a better position to decide on the terms and scope of any subsequent contract.
Recommendation 27: Don't blindly call up specifications in the contract without first understanding the impact. Recognise that standards are generally written as general documents that may require some level of tailoring to match with the specific project aims.

Specifications a written as general documents aimed at covering all circumstances within a particular (possibly narrow) field. They need to be examined carefully to determine the appropriateness for the particular development planned.

Recommendation 28: Information Technology practitioners should be made aware that silence itself may constitute a breach of Section 52 of the Trade Practices Act, 1974 as conduct which is misleading or deceptive (or at least conduct which was likely to mislead or deceive).

If a group or team is formed consisting of representatives from different contracted parties, it is important to be active in the decision making process and in particular, raise clear objections to group decisions where appropriate.

Active participation in teams is important because it is necessary to give clear messages to the other parties if legal claims such as misrepresentation, estoppel and misleading and deceptive conduct are to be avoided.

4.9.9 Conclusion
Overall in this case study it appears that a fair result was achieved for the Developer. The Systems Integrator, on the other hand, ended up with a system which had a greater number of defects than the contract originally anticipated. In order to avoid this situation the Systems Integrator should have firstly been more aware of the development standard called for in the contract and possibly tailored it appropriately before entering into the contract; and secondly, been a more active member of the Problem Report Review Group in rejecting and decisions of that group with which it did not agree.

Quality assurance is well recognised as a cornerstone of many endeavours and information technology activities are no exception. It is thus recommended (among other things) that quality assurance activities be given contractual weight where possible.
4.10 CASE STUDY 6 - MICRO-AGREEMENTS IN THE CONTEXT OF SOFTWARE CONFIGURATION MANAGEMENT

4.10.1 Introduction and Explanation of the Activity

One of the most important activities associated with any software development is configuration management. This activity is intended to ensure that, among other things, different versions of the software and related documentation are "sealed" in a protective environment so that firstly, no changes can be made unless they are done so under controlled conditions; and secondly, so that particular versions of the software can be found, identified and issued to customers as required. Any large development involves significant amounts of change and configuration management activities ensure that this change is made in a formally controlled manner. The process of configuration management has been described in the following terms:

"Software Configuration Management involves selecting project baseline items (e.g. the project description, products, and process specifications of the project), controlling these items and changes to them, and recording and reporting status and change activity for these items. Changes to these baseline items are controlled systematically using a defined change control process. The configuration (software or documentation) of a system, or of any of the controlled intermediate or support products, can be distinctly identified at any point in time."\(^{487}\)

4.10.2 Example of a Micro-agreement

A document was written to describe the technical details of an interface between two software programs (Program A and Program B) which constituted part of a banking system. Subcontractor A was engaged by the Systems Integrator to produce Program A, while Program B was to be developed by the Systems Integrator itself. As the two programs needed to communicate with each other, each needed to conform exactly with the published interface specification.

The interface specification document itself was written by the Systems Integrator on a "UNIX\(^{488}\) computer platform and placed in a protected area. The specification was written in a computer programming language called "Ada". Ada has a facility where "specifications" can be written and checked by compilation before the full program is written. Overall this means that the specification produced should have been complete and syntactically correct (as far as possible) at the time of issue.


488 UNIX is a particular type of "operating system". An "operating system" is "a collection of software, firmware, and hardware elements that controls the execution of computer programs and provides such services as computer resource allocation, job control, input/output control, and file management in a computer system". IEEE Standard 610.12-1990 IEEE Standard Glossary of Software Engineering Terminology.

Paul P. Parnell 161
When the interface specification was provided to Subcontractor A however a micro-agreement was made between representatives of the Systems Integrator and Subcontractor A whereby the configured UNIX file was copied to a Microsoft Word file and issued both as a paper printout of the Word file as well as an electronic copy of the Word file on a DOS\textsuperscript{489} floppy disk.

A problem occurred because the DOS file and the UNIX file contained different formatting and control characters. Visual inspection of the files on the respective DOS and UNIX computer screens revealed no significant difference between the two files however the hidden control characters prevented the specification from being compilable under the Ada software language used. Additional work was required to identify why the specification as provided could not be compiled. Once the problem was found, another copy of the specification was provided in the correct file format.

4.10.3 Potential Problems arising from the Micro-agreement

As a result of the micro-agreement:

1. The two software programs were incompatible because the interfaces of each did not match exactly.

2. Additional work needed to be performed by the Systems Integrator to rewrite its software program to match with the program from Subcontractor A.

4.10.4 Avoiding the Problem

The situation described in this Case Study was caused by ineffective configuration management practices. A version of a protected item (i.e. the interface between the two programs) was inadvertently allowed to be modified and issued. The modification was not intentional and in fact was not even obvious at the time - simply proving the specification in electronic format but in a different file format than that originally developed created the problem.

In order to avoid a situation of this type it is recommended that contractual weight be given to enforcing a solid configuration management process. This can be done in a number of ways including requiring conformance to relevant standards and procedures.

A number of quality related standards such as IEEE STD 1298/A3563.1, Software Quality Management System; and ISO 9000-3, Guidelines for the Application of ISO 9000 to the Development, Supply, and Maintenance of Software provide support for configuration management activities although in large developments it is recommended, where possible, that a configuration management plan be written and reviewed by the buyer.

\textsuperscript{489} DOS is a proprietary operating system designed for use mainly on personal computers.
before an implementation contract is signed. A number of the documents listed in Appendix A. Software Standards relate to configuration management and it is recommended that these (and others) be reviewed with the aim of incorporation into Systems Integration and software development contracts. As an example, the following may be relevant to a particular development:

- ANSI/IEEE Std 1042, Guide to Software Configuration Management
- MIL-STD-973, Configuration Management
- MIL-HDBK-61, Guidelines for Configuration Management

Regardless of which standards are used, it is generally recognised within the software engineering community that certain activities need to be performed in order to provide an effective configuration management function. As such, these activities should be incorporated into a contract (where appropriate) either explicitly or by reference. On example of a list of Configuration Management activities can be found in the Software Engineering Capability Maturity Model\(^{490}\). These activities are described briefly below.

- Different levels of configuration management should be implemented, as appropriate, during the project’s life cycle. A configuration management process needs to be flexible enough to cope with the different products associated with a particular development; and in addition, also may need to vary over the life of a product. For example, more change is usually expected during the early stages of development than towards the end. The amount of control needed to support these changes may thus be different during the early stages than later on, and the configuration management process needs to be able to apply the relevant degree of flexibility. Overly rigid control may stifle the developers and be expensive to implement, while inadequate control may cause problems such as those identified in this Case Study.

- A documented software configuration management plan should exist, and this should be used as the basis for configuration management activities. Ideally this should be available for review before a contract is in place. In larger projects consideration should be given to an initial preparation contract for development of this and other related documents. Organisations which have been accredited to a quality standard such as IEEE STD 1298/A3563.1, Software Quality Management System, (for example) will probably have a configuration management process already in place, although ideally this should be assessed for adequacy under the particular circumstances of each development.

- A configuration management library system should be established as a repository for the software and documentation baselines.

- The items to be placed under formal configuration control should be identified early in the development, and preferably in the Configuration Management Plan.

- A documented procedure should be followed for initiating, recording, reviewing, approving, controlling and tracking changes to all configured items.

- A documented procedure should be followed to create and control the release of configured items. The problem identified in this Case Study may have been avoided if this activity was undertaken effectively.

• A documented procedure should be followed to record the status of configured items. At any time it should be possible to establish the status of configured items and identify approved changes which have been made to the item over time.

• Standard reports documenting the configuration management activities should be created and distributed for review.

• A documented procedure should be followed to prepare for, conduct, report results from, and track actions from software baseline audits.

To ensure that an effective configuration management function is in place, the above activities need to be undertaken and preferably incorporated into a contract. The incorporation can be either explicitly or by reference to appropriate standards, procedures and guidelines.

4.10.5 Legal Analysis of Problems
This case study provides a simple example where inadequate configuration management practices allowed inappropriate changes to be made to a software product before delivery to a subcontractor for integration. In this particular instance the problem, once identified, was fairly easy to fix. Hence the amount of "damage" was minimal in this case study as a result of the micro-agreement. Nonetheless the following action may be available.

<table>
<thead>
<tr>
<th>Action</th>
<th>Plaintiff</th>
<th>Defendant</th>
<th>Claim</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantum Meruit</td>
<td>Sub-Contractor A</td>
<td>System Integrator</td>
<td>Claim is for additional costs of modifying the software to allow for the different file formats</td>
</tr>
</tbody>
</table>

A general discussion on the action of quantum meruit has been provided above (see Section 3.5 above) and in substance can be found if one party was unjustly enriched at the expense of the other, provided the work done was not of the kind that is normally undertaken gratuitously. Once unjust enrichment has been found, then a claim in quantum meruit involves "the payment of an amount which constitutes, in all relevant circumstances, fair and just compensation for the benefit or 'enrichment' actually or constructively accepted".

In the present case study if this action was successful, then the amount which could be claimed is likely to correspond to "remuneration calculated at a reasonable rate for work actually done" provided that the amount is so minimal that "the claim ... is not a frivolous or vexatious one and accordingly an abuse of the process of the Court".

It is arguable whether a contract actually existed which covered the additional work needed to be performed to rectify the discrepancies in the interface specification. It is however not necessary to show that a contract did exist in order to found a claim in quantum meruit. The action is based on the doctrine of unjust enrichment and "the right to recover does not depend upon the existence of an implied contract but on a claim to restitution based upon unjust enrichment".

491 see Craven-Ellis v Canons Ltd (1936) 2 KB 403
492 Pavey and Mathews Pty Ltd v Paul (1987) 162 CLR 221 at 263-264
493 Allstate Life Insurance Co. and Others v Australian and New Zealand Banking Group Limited and Others No. G381 of 1994 FED No. 45/95, Beaumont J. at para 18
494 Pavey and Mathews Pty Ltd v Paul (1987) 162 CLR 221; Australian & New Zealand Banking Group Ltd v Westpac Banking Corporation (1986) 62 ALJR 292

Paul P. Parnell 164
In summary, it is submitted that a claim may be available by Sub-Contractor A against the System Integrator for an action in quantum meruit for additional work required to correct discrepancies between the various copies of the interface specification. This of course would be subject to the requirement that the claim was not frivolous or vexatious and accordingly some clear evidence would need to be available to show that the amount of rework is significant in that regard.

Of course the actual dollar amount of any claim would be subject to scrutiny by a court - solid evidence of the actual costs associated with undertaking the work would naturally help a parties case. Accordingly good record keeping is essential for any organisation, and particularly where evidence may need to be provided to a court.

In this case study the responsibility for the problem rested with the Systems Integrator (on the facts given) in that it appeared to provide to the sub-contractor a “corrupted” data file. It may have been the case in this instance that neither party was familiar with good configuration management practices and accordingly, an express term in the contract which clearly identified the responsibility for this function may have helped to avoid the problem.

4.10.6 Industry Code of Conduct
A discussion of a proposed industry code of conduct: Information Technology Systems Integration and Outsourcing Code of Conduct is provided in Section 4.11. The discussion provided in that Section addresses issues and makes recommendations for the disclosure of software configuration management practices in the information technology systems integration and outsourcing industries.

4.10.7 Findings
In relation to this Case Study key findings are summarised below.

Finding 27: A micro-agreement existed in this case study. (Specifically, a micro-agreement was made between representatives of the Systems Integrator and a Subcontractor to issue a computer file in a different format than originally configured).

The micro-agreement consisted of an arrangement between representatives of the Systems Integrator and Subcontractor A to issue a configured computer file in an incompatible file format.

Finding 28: The micro-agreement had a significant impact on the development. (Specifically, two software programs were incompatible because the interfaces of each did not match. Additional work was needed to be performed by the Systems Integrator because of this incompatibility).

The impact of the micro-agreement in this case study was firstly, the two software programs were incompatible because the interfaces of each did not match; and secondly, additional work needed to be performed by the Systems Integrator to rewrite its software program to match with the program from Subcontractor A.
Finding 29: The micro-agreement needed to be carefully managed to minimise risk and maximise benefits.

Recommendations on how to minimise the risks and maximise the benefits are provided in the following Section.

4.10.8 Recommendations

Overall, in order to avoid or minimise the problems the following recommendations are made.

Recommendation 30: Link the contract to Software Engineering "best practice".

Ensuring that a development is undertaken using software engineering best practice is important. What exactly "best practice" is depends on the particular development and the technologies used. The following recommendation gives an example of one such best practice which should be undertaken in a formal sense in any significant systems integration or outsourcing project.

Recommendation 31: Provide support in the contract for an effective Configuration Management process.

Configuration Management provides the necessary controls to ensure (among other things) that the exact state of developed products is known. It ensures that no confusion or loss of data can occur through the ongoing change of software products by multiple parties, and in addition ensures that what the customer receives is what it is supposed to have received.

Recommendation 32: A voluntary "Industry Code of Conduct" should be developed in accordance with the Trade Practices Act, 1974 Part IVB to ensure a Systems Integrator or Outsourcing organisation discloses to the customer the Configuration Management activities to be undertaken.

The key concept here is disclosure. If a full and frank disclosure of the Configuration Management activities to be undertaken by the systems integrator or outsourcer is made, then all parties should be in a better position to decide on the terms and scope of any subsequent contract.

Recommendation 33: Detailed records should be maintained within an organisation to show actual costs of work. Such records may be used as evidence to help in the assessment of quantum.
Recommendation 34: Detailed records should be maintained within an organisation of interactions between parties, including micro-agreements. Such records may be used as evidence to help in the assessment of liability.

The above two recommendations are designed to provide evidence, if required, in the event of litigation.

4.10.9 Conclusion

In summary, this case study has shown how inadequate control of configuration activities can cause problems during software development and integration activities. Although, in this particular case study, the "damage" caused was not particularly great, configuration management is generally recognised as an important activity which if not undertaken has potential to cause serious harm.

In conclusion, configuration management is an activity which is recognised within the information technology industry as being fundamental to the successful outcome of a development. This case study has provided an example of where inadequate configuration management can cause problems, and recommends, among other things, giving contractual weight to these activities as appropriate.
4.11 INFORMATION TECHNOLOGY INDUSTRY CODE OF CONDUCT

4.11.1 Issues of Disclosure

The case studies presented in the preceding Sections have identified some of the more important issues to be addressed in information technology projects. These issues include the fundamental software engineering practices: requirements management, software project planning, software project tracking and oversight, subcontractor management, software quality assurance, and software configuration management.

It is recognised that in real-world developments however that particular acquirers of software systems may not always require such fundamental practices to be in place. Reasons for this vary but are probably either driven by cost or schedule considerations, or are the result of ignorance of the benefits of such practices. If however the acquirer has notification or disclosure of the level of activity in this particular process area then it is more likely to make appropriate procurement decisions. One way of obtaining this disclosure is through an industry code of conduct.

4.11.2 The Legal Basis for an Industry Code of Conduct

The Trade Practices Act, 1974 Part IVB provides for the legal recognition of industry codes of conduct where that term (i.e. industry code) means "a code regulating the conduct of participants in an industry towards other participants in the industry or towards consumers in the industry". Under the Trade Practices Act, 1974, industry codes can be prescribed by regulation and defined as mandatory or voluntary codes.

In terms of remedies, the trade Practices Act, 1974 gives force to an industry code through remedies under Section 51AD including:

- **Injunction**: A person may seek an injunction under s.80 to restrain breaches of the prescribed provisions.

- **Damages**: A person who suffers loss or damage by conduct in breach of the code may seek to recover the loss or damage by action against that other person under s.82 of the Trade Practices Act.

- **Undertakings**: The Australian Competition and Consumer Commission (ACCC) may accept enforceable written undertakings under s.87B of the Trade Practices Act in relation to matters covered by the code.

- **Corrective Advertising**: The Minister or the Commission may seek a court order to disclose information or publish corrective advertising under s.80A of the Trade Practices Act in relation to matters under this code.

- **Other orders**: A Court may make such orders under s.87 as it thinks appropriate against a person who has been involved in a contravention if it considers that the orders will compensate for loss or damage suffered as a result of the breach.

465 Trade Practices Act, 1974, Section: 51ACA(1)
466 Trade Practices Act, 1974, Section: 51AE(a)
467 Trade Practices Act, 1974, Section: 51AE(b)
468 Department of Workplace Relations and Small Business: Franchising Code of Conduct, 1 July 1998, Preamble
Significant legal weight is can thus be given to industry codes under appropriate circumstances.

4.11.3 Existing Codes of Conduct

In relation to the issue of disclosure, an example of an industry code which includes obligations of disclosure, is the Franchising Code of Conduct. Part 2 of that code of conduct is titled "Disclosure" and sets out the requirements for disclosing certain facts applicable to the franchising context. Specifically, it requires a disclosure document to be produced and issued to interested parties to a franchising transaction. The code dictates the format of the disclosure document as well as a list of items which must be included. That list includes many items to be disclosed including:

- Franchisor details
- Litigation (pending or in progress)
- Existing franchises
- Intellectual property
- Franchise territory
- Supply of goods or services to a franchisee
- Supply of goods or services by a franchisee
- Marketing or other co-operative funds
- Franchisor's obligations
- Franchisee's obligations
- Earnings information
- Financial details

4.11.4 Advantages and Disadvantages of Codes of Conduct

Generally speaking, some advantages of having a code of conduct in place include:

- encouraging fair dealing between suppliers and buyers,
- improving communication between suppliers and buyers,
- providing buyers of with some degree of visibility of the processes used by suppliers so that it can make a more accurate assessments of risk,
- providing a balanced basis for comparing between different suppliers,
- providing a framework for structured dispute resolution,
- providing remedies for breaches of the code including damages, injunctions and other remedies allowed under the Trade Practices Act, 1974, and
- providing a base standard of conduct which provides more certainty in commercial transactions.

Despite these advantages, a number of disadvantages may also exist which include:

- a code of conduct by necessity is likely to provide general rules and accordingly may not be entirely appropriate in all specific circumstances,
- having a code of conduct may provide some false sense of security to consumers in believing that they may have protection in circumstances where this may not be the case,
- conforming to obligations required by a code may add additional and sometimes unnecessary cost and schedule implications to transactions,

---

499 Department of Workplace Relations and Small Business: Franchising Code of Conduct, 1 July 1998,
500 Department of Workplace Relations and Small Business: Franchising Code of Conduct, 1 July 1998, Annexure 1: Disclosure Document for Franchisee or Prospective Franchisee

Paul P. Parnell
169
• in relation to codes of conduct covering technical issues, the requirements of the code may become superseded and need updating on a regular basis as the technology evolves
• the terminology "code of conduct" may imply lesser obligations than terms such as "law" or "contract" when in fact a code of conduct under the Trade Practices Act, 1974 does provide for strict legal rights and obligations. This needs to be made clear to practitioners and groups who sign up to a code of conduct.

In the implementation of any code of conduct, the above issues need to be taken into account and addressed in a balanced fashion to ensure that benefits overall outweigh the 'costs' associated with setting up, administering, and enforcing compliance.

4.11.5 Information Technology Systems Integration and Outsourcing Code of Conduct

It is submitted that a degree of disclosure may also be appropriate to the software industry, particularly in areas where traditionally there is a high degree of risk or uncertainty. For software based systems requiring significant development, integration, or outsourced work, it is such disclosure should ideally include the processes used, specifically in the context of software engineering fundamental practices.

4.11.5.1 Primary Aims

The primary aims and desired outcomes of the (proposed) Information Technology Systems Integration and Outsourcing Code of Conduct are:

• to encourage fair dealing between suppliers and buyers of information technology products and services, within the context of existing trade practices laws
• to improve communication and strengthen relationships between suppliers and buyers of information technology products and services
• to provide buyers of with some degree of visibility of the processes used by suppliers so that it can make a more accurate assessments of risk
• to provide a balanced basis for comparing between different suppliers and outsourcers.

4.11.5.2 Outline of Proposed Code of Conduct

As a starting point, it is submitted that the (proposed) Information Technology Systems Integration and Outsourcing Code of Conduct should include the elements identified below. It should be noted that the outline provided below is not intended to provide a complete code of conduct but rather to highlight one way in which software engineering best practices could be introduced into the industry.
[PROPOSED] INFORMATION TECHNOLOGY SYSTEMS INTEGRATION AND OUTSOURCING CODE OF CONDUCT

Preamble

Part 1: Preliminary

Name of code
INFORMATION TECHNOLOGY SYSTEMS INTEGRATION AND OUTSOURCING CODE OF CONDUCT

Purpose of code
(The purpose of the Code should be stated here. This Section should identify the overall aims as stated above in Section: 4.11.5.1)

Application
(The code should apply to suppliers and buyers of information technology products and services, initially on a non-mandatory basis)

Information Technology Agreement
(This Section provides a definition of agreements to which this code applies. In general it should cover agreements associated with the acquisition of information technology products and services involving significant systems integration or outsourced elements.)

Terminology and Definitions
(This Section includes relevant definitions)

Compliance
(This Section should provide general information about compliance. If the code is implemented as a non-mandatory code it should be made clear that despite being non-mandatory, the Trade Practices Act, 1974 may still apply in relation to relevant issues such as misleading and deceptive conduct)

Dispute resolution
(Identification of dispute resolution process (if any) should be included)

Enforcement
(A brief discussion of the process for enforcement of obligation should be made and the relevant parties such as the Australian Competition and Consumer Commission (ACCC)

Remedies for breach
(A summary of possible remedies for breach of the code should be included here. It should be made clear that even if the code is not mandatory, certain remedies may be available for conduct which breaches the Trade Practices Act, 1974 and related laws.)

Amendment of the Code
(This Section identifies who is responsible for amendments to the code)

The Information Technology Code of Conduct Policy Council
(An independent body should be established to provide advice on policy issues related to the code of conduct. This body should include representation of interested stakeholders including information technology buyer and supplier organisations, professional bodies such as the Association of Engineers, Australia and the Australian Computer Society; as well as end-users of computer based products)

Point of contact

Part 2: Disclosure

Division 2.1 Disclosure document
(This division should specify the requirements for a disclosure document including whether it is mandatory or not and also identify the recommended content and layout of such a document)

Part 3: Conditions of Information Technology Agreement
(This part is not intended to be the actual contract between the buyer and the seller but rather to identify any particular conduct that is desirable from the industry perspective. For example, this Section may include: requirements to give reasonable notice and
Part 4: Resolving disputes
(This part should provide a recommended process for the resolution of disputes associated with the code)

Annex - Disclosure Document Outline

1. Cover page
2. Confidentiality of this document
   (Identifies whether any information in this disclosure document includes trade secrets or other intellectual property and the obligation on the recipient in that regard)
3. Information technology supplier details
4. Information technology supplier experience
   (This Section is intended to identify the extent and type of information technology work previously conducted)
5. Software engineering practices used by the supplier including:
   - Requirements management practices
   - Software project planning practices
   - Software project tracking and oversight practices
   - Subcontractor management practices
   - Software quality assurance practices
   - Software configuration management practices
6. Litigation
   (This Section identifies relevant areas of past or present dispute involving the supplier)
7. Intellectual property
   (This Section identifies any relevant intellectual property issues in relation to the products and services to be supplied including such things as: whether the supplier owns relevant intellectual property, or if not, who owns it; whether there is any judgement or pending proceedings in relation to relevant intellectual property; whether there is any agreement in place which affects relevant intellectual property; whether there are any limitations on the use of relevant intellectual property)
8. Financial information
   (Disclosure of the financial "health" of the supplier is provided to enable an assessment of risk in that regard)
9. Suppliers obligations
   (This Section lists the obligations of the Supplier under this industry code of conduct)
10. Buyers obligations
    (This Section lists the obligations of the Supplier under this industry code of conduct)
11. Other relevant disclosure information

Figure 26: Outline of [Proposed] "Information Technology Systems Integration and Outsourcing Code of Conduct"

The above outline is based on the Franchising Code of Conduct\textsuperscript{501} however the main point to note is that disclosure of certain software engineering best practises is included. The disclosure of such information will impact the buyer and supplier of software based products and services in different ways.

From the buyers' perspective, such disclosure is likely to give some indication of the capability of the supplier in the provision of complex information technology products and services. This information can then be used to make an assessment of the risk associated with the acquisition. The provision of such information itself does not come at no cost however and as a consequence, the buyer (directly or indirectly) may have to pay for the development of the disclosure documentation provided under the code of conduct.

\textsuperscript{501} Department of Workplace Relations and Small Business: Franchising Code of Conduct, 1 July 1998

Paul P. Parnell

172
From the suppliers perspective it is suspected that some suppliers may be reluctant to disclose such information, particularly if it discloses some “deficiencies” in its software engineering process, or if the information is considered to be confidential to the supplier. In order to address these concerns it is likely that a number of mechanisms will need to be put in place such as confidentiality agreements. Despite this, current industry “culture” generally recognises that software engineering capability can be measured and accordingly used as a marketing tool itself\textsuperscript{502}.

In the first instance, it is recommended that the proposed code of conduct be voluntary rather than mandatory. The reason for this is that sufficient time should be given to trial the new code before its use is forced upon situations where it may not be entirely appropriate.

Overall, it is generally recognised in the software engineering community that the use of certain fundamental best practises can provide a better outcome in terms of cost, schedule and quality\textsuperscript{503}. Although it may not be possible (or advisable) to mandate the use of such best practices in all software development activities, it is suggested that some disclosure of the extent to which they are used would be useful. The development of an “Information Technology Systems Integration and Outsourcing Code of Conduct” should allow parties to make better informed decisions about software acquisition; raise the level of awareness of the fundamental best practice areas within the software industry; and encourage fair dealing between suppliers and buyers of information technology products and services, within the context of existing trade practices (and related) laws.

4.12 SUMMARY OF FINDINGS

4.12.1 Introduction

The case studies provided in this chapter are intended to describe situations where “micro-agreements” have a significant impact on the outcome on software development activities generally, and systems integration and outsourcing work in particular.

As the number of possible software developments is probably infinite, a conscious attempt has been made to provide a consolidated set of case studies which provide a model for analysis of the most important activities. The set of “most important” activities was derived from information technology literature, with the end result being a defined set of activities generally acknowledged within that industry as being fundamental to the achievement of a solid software engineering capability.

In summary the six fundamental areas of software engineering activity covered by the case studies are:

- Case Study 1: Requirements management
- Case Study 2: Software project planning
- Case Study 3: Software project tracking and oversight
- Case Study 4: Subcontractor management
- Case Study 5: Software quality assurance
- Case Study 6: Software configuration management

\textsuperscript{502} For example, if a corporation can claim that it has accreditation to a software capability standard or model (such the Capability Maturity Model – see Bibliography) then it may have some marketing advantage over other organisations which do not.

Each of the case studies was analysed in detail, starting with an identification of the micro-agreement itself and the problems caused by it. An analysis was then conducted to find firstly, how the problems could be prevented from occurring in the first place; and secondly, what legal options are available for resolving issues after the problems had occurred.

A summary of findings from this analysis are listed below, firstly in the context of each case study, and then collectively for all case studies. Subsequently, a summary of recommendations is provided in a later Section which link to the findings.

4.12.2 List of Case Study Findings

This list of findings is extracted from the case studies in this chapter. Note that although some of the findings listed below are worded similarly between case studies, the reasons for each finding are different. Accordingly, please refer to the particular case study for a discussion of the relevant issues associated with each finding.

<table>
<thead>
<tr>
<th>Case Study 1 - Requirements Management (See Section: 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finding 1:</strong> A micro-agreement existed in this case study. (Specifically, the micro-agreement consisted of an agreement between employees of two companies to implement a slightly modified interface rather than the one specified in the prime contract).</td>
</tr>
<tr>
<td><strong>Finding 2:</strong> The micro-agreement had a significant impact on the development. (Specifically, the end result of the micro-agreement was that three of the sub-contractors needed to perform additional work at increased cost and schedule).</td>
</tr>
<tr>
<td><strong>Finding 3:</strong> The micro-agreement was highly likely to occur during the conduct of the development because of the inherent nature and size of the project and the lack of formal Requirements Management activity.</td>
</tr>
<tr>
<td><strong>Finding 4:</strong> The micro-agreement was an important form of communication between the parties, and would have resulted in a product of higher quality with reduced cost and schedule. Accordingly, micro-agreements should not be prevented from occurring but rather allowed to occur under controlled conditions.</td>
</tr>
<tr>
<td><strong>Finding 5:</strong> The information technology agreement in this case study did not adequately take the effect of the micro-agreement into account.</td>
</tr>
<tr>
<td><strong>Finding 6:</strong> The project needed to be carefully managed to minimise risk and maximise benefits of the micro-agreement. (Refer Section: 0 and following for specific details and Section: 4.13.2 for a summary of recommendations).</td>
</tr>
<tr>
<td><strong>Finding 7:</strong> Where an opinion is given or a prediction is made by an expert in the relevant field, then it could be misleading if the expert did not in fact use his or her expertise, or did not otherwise have rational grounds for expressing the opinion or making the prediction.</td>
</tr>
</tbody>
</table>

Paul P. Parnell
<table>
<thead>
<tr>
<th>Finding 8: Conduct and statements made during the course of pre-contractual negotiations can lead to binding obligations and an award of damages, even if the parties have not entered into a contract</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finding 9: Information technology custom or trade usage may be implied into a contract. Custom or trade usage within the IT industry may include expectations of having fundamental software engineering practices in place such as quality assurance systems.</td>
</tr>
<tr>
<td>Case Study 2 - Software Project Planning (See Section: 4.6)</td>
</tr>
<tr>
<td>Finding 10: A micro-agreement existed in this case study. (Specifically, the micro-agreement made was explicitly modify the contractual &quot;lifecycle&quot; process as defined in the main contract and build a working software prototype before the contracted requirements and design documentation were complete).</td>
</tr>
<tr>
<td>Finding 11: The micro-agreement had a significant impact on the development. (Specifically, the product was delivered late).</td>
</tr>
<tr>
<td>Finding 12: The micro-agreement needed to be carefully managed to minimise risk and maximise benefits. (Refer Section: and following for specific details and Section: 4.13.2 for a summary of recommendations).</td>
</tr>
<tr>
<td>Case Study 3 - Software Project tracking and Oversight (See Section: 4.7)</td>
</tr>
<tr>
<td>Finding 13: A micro-agreement existed in this case study. (Specifically, two of the parties involved agreed that approximately 25% of the requirements could be classified as &quot;non-essential&quot;, although still required).</td>
</tr>
<tr>
<td>Finding 14: The micro-agreement had a significant impact on the development. (Specifically, the product was difficult for the users to operate under normal operational conditions. Additional work needed to be performed under a separate contract at additional cost).</td>
</tr>
<tr>
<td>Finding 15: The micro-agreement needed to be carefully managed to minimise risk and maximise benefits. (Refer Section: 4.7 and following for specific details and Section: 4.13.2 for a summary of recommendations).</td>
</tr>
<tr>
<td>Finding 16: This case study provides an example of where the Buyer of a system, through its actions of affirming the contract (partially because of business pressure), may have no suitable remedy at law.</td>
</tr>
<tr>
<td>Finding 17: The legal concept of &quot;reasonable time&quot; may be difficult to establish in the context of large systems integration contracts because firstly, the number of such contracts is relatively few; secondly, published data on the actual time for performance may be difficult and costly to obtain due to the proprietary nature of such information; and thirdly, the time for performance of contractual obligations in a systems integration contract will be somewhat driven by the technology of the day.</td>
</tr>
<tr>
<td>Case Study 4 - Subcontractor Management (See Section: 4.8)</td>
</tr>
</tbody>
</table>

Paul P. Pamell
| Finding 18: | A micro-agreement existed in this case study. (Specifically, the micro-agreement consisted of an agreement between the Systems Integrator and Subcontractor B that Subcontractor B could use preliminary performance estimates in the design of its sub-system). |
| Finding 19: | The micro-agreement had a significant impact on the development. (Specifically, the Systems Integrator incurred additional cost in upgrading hardware to run the sub-system developed by a subcontractor). |
| Finding 20: | Cases examining computer system developments have effectively elevated "mere puffs" to the position of misleading and deceptive conduct under Section 52 of the Trade Practice Act, 1974. As a consequence of the law in this regard, predicted computer system performance estimates should not be provided unless they are accurate and correct, or unless heavily caveated so as to cause no misunderstanding. |
| Finding 21: | The micro-agreement needed to be carefully managed to minimise risk and maximise benefits. (Refer Section: and following for specific details and Section: 4.13.2 for a summary of recommendations). |
| Finding 22: | Attempting to exclude performance figures from a contract per se may avoid contractual liability, although if pre-contractual claims are made which are misleading or deceptive then an action may be available through other means such as the Trade Practices Act, 1974. In addition, a failure by a supplier to expressly deny that it can meet performance criteria required by the Buyer can be sufficient grounds to found an action under Section 52 of the Trade Practices Act, 1974. |

**Case Study 5 - Software Quality Assurance (See Section: 4.9)**

| Finding 23: | A micro-agreement existed in this case study. (Specifically, A micro-agreement was made between the members of the "Problem Report Review Group" to change the "Problem Classification" definitions originally defined in standards called up in the contract). |
| Finding 24: | The micro-agreement had a significant impact on the development. (Specifically, the final system was delivered with a greater number of defects than necessary. Additional work was needed to rectify these problems. A separate contract at additional cost was put in place to do this). |
| Finding 25: | The micro-agreement needed to be carefully managed to minimise risk and maximise benefits. (Refer Section: and following for specific details and Section: 4.13.2 for a summary of recommendations). |
| Finding 26: | Silence itself may constitute a breach of Section 52 of the Trade Practices Act, 1974 as conduct which is misleading or deceptive (or at least conduct which was likely to mislead or deceive). |

**Case Study 6 - Software Configuration Management (See Section: 4.10)**
Finding 27: A micro-agreement existed in this case study. (Specifically, a micro-agreement was made between representatives of the Systems Integrator and a Subcontractor to issue a computer file in a different format than originally configured).

Finding 28: The micro-agreement had a significant impact on the development. (Specifically, two software programs were incompatible because the interfaces of each did not match. Additional work was needed to be performed by the Systems Integrator because of this incompatibility).

Finding 29: The micro-agreement needed to be carefully managed to minimise risk and maximise benefits. (Refer Section: 4.10 and following for specific details, and Section: 4.13.2 for a summary of recommendations).

4.12.3 Consolidated Findings
It can be seen from the above table that some of the findings between case studies have similar or identical wording. Although the reasons for each finding are different depending upon the particular facts of each case study, a consolidated list which attempts to identify the core findings is provided below:

- Micro-agreements do exist
- micro-agreements can have a significant impact on developments
- micro-agreements need to be carefully managed to minimise risk and maximise benefits
- software engineering “best practice” can provide one way of controlling micro-agreements
- linking software engineering “best practice” to the contract will help to minimise the risk and maximise the benefits of micro-agreements
- Standards provide one way of linking software engineering “best practice” to a contract
- Development of an industry Code of Conduct consistent with the Trade Practices Act, 1974, Section 51AC may be useful in providing sufficient disclosure to contracting parties on what fundamental software engineering best practice activities are proposed to be undertaken.

Based on the above findings, the following recommendations are made.

4.13 SUMMARY OF RECOMMENDATIONS

4.13.1 Introduction
In line with these findings, it is submitted that one way of dealing with micro-agreements is to use legal devices which are designed to encourage interaction and innovation between the parties. These devices should be made available through general law or enforced contractually. Specifically, the recommendations made in each case study are list below individually, and in the following Section, collectively.

4.13.2 List of Case Study Recommendations
This list of recommendations is extracted from the case studies in this chapter. Note that although some of the recommendations listed below are worded similarly between case
studies, each recommendation is based on a different set of facts peculiar to the relevant case study. Accordingly, please refer to the particular case study for a discussion of the relevant issues associated with each recommendation.

**Case Study 1 - Requirements Management (See Section: 0)**

<table>
<thead>
<tr>
<th>Recommendation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Link the contract to Software Engineering “best practice”.</td>
</tr>
<tr>
<td>2</td>
<td>Provide support for an effective Requirements Management Process in the Contract.</td>
</tr>
<tr>
<td>3</td>
<td>Micro-agreements should not be prevented from occurring but rather allowed to occur under controlled conditions.</td>
</tr>
<tr>
<td>4</td>
<td>The contractual doctrine of an “implied duty to co-operate” should be modified to explicitly apply “throughout the period of the contract, to the extent that it is necessary to make the contract workable”.</td>
</tr>
<tr>
<td>5</td>
<td>As a strategy for avoiding a potential action under Section 52 of the Trade Practices Act, 1974 IT practitioners should be made aware that where an opinion is given or a prediction is made by an expert in the relevant field, then it could be misleading if the expert did not in fact use his or her expertise, or did not otherwise have rational grounds for expressing the opinion or making the prediction.</td>
</tr>
<tr>
<td>6</td>
<td>As a strategy for avoiding a potential action under the principals of estoppel, IT practitioners should be made aware that conduct and statements made during the course of pre-contractual negotiations can lead to binding obligations and an award of damages and other remedies, even if the parties have not entered into a contract.</td>
</tr>
<tr>
<td>7</td>
<td>As a strategy for avoiding any unintended consequences due to terms being implied into a contract, IT practitioners should be made aware that terms related to custom or trade usage may be implied into a contract. Custom or trade usage within the IT industry may include expectations of having fundamental software engineering practices in place such as quality assurance systems. Where there is any doubt terms should be expressed in contracts.</td>
</tr>
<tr>
<td>8</td>
<td>A voluntary “Industry Code of Conduct” should be developed in accordance with the Trade Practices Act, 1974 Section 51AC to ensure a Systems Integrator or Outsourcing organisation discloses to the customer the Requirements Management activities to be undertaken.</td>
</tr>
</tbody>
</table>

**Case Study 2 - Software Project Planning (See Section: 4.6)**

<table>
<thead>
<tr>
<th>Recommendation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>Link the contract to Software Engineering “best practice”.</td>
</tr>
<tr>
<td>Recommendation 10:</td>
<td>Provide support for effective project planning in the contract.</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>Recommendation 11:</td>
<td>Ensure that an appropriate software development life-cycle model is used.</td>
</tr>
<tr>
<td>Recommendation 12:</td>
<td>A voluntary “Industry Code of Conduct” should be developed in accordance with the Trade Practices Act, 1974 Section 51AC to ensure a Systems Integrator or Outsourcing organisation discloses to the customer the Project Planning activities to be undertaken.</td>
</tr>
</tbody>
</table>

**Case Study 3 - Software Project tracking and Oversight (See Section: 4.7)**

<table>
<thead>
<tr>
<th>Recommendation 13:</th>
<th>Link the contract to Software Engineering &quot;best practice&quot;.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recommendation 14:</td>
<td>Provide support for tracking and oversight activities in the contract.</td>
</tr>
<tr>
<td>Recommendation 15:</td>
<td>This case study provides an example of where the Buyer of a system, through its actions of affirming the contract (partially because of business pressure), may have no suitable remedy at law. Accordingly it is recommended that IT practitioners be made aware that vigilance in commercial transactions is essential to maintain legal rights. In this case study a micro-agreement made between a Buyer and a software developer affected the rights of that Buyer under a separate contract between itself and the Systems Integrator.</td>
</tr>
<tr>
<td>Recommendation 16:</td>
<td>A voluntary “Industry Code of Conduct” should be developed in accordance with the Trade Practices Act, 1974 Section 51AC to ensure a Systems Integrator or Outsourcing organisation discloses to the customer the tracking and oversight activities to be undertaken.</td>
</tr>
</tbody>
</table>

**Case Study 4 - Subcontractor Management (See Section: 4.8)**

<table>
<thead>
<tr>
<th>Recommendation 17:</th>
<th>Link the contract to Software Engineering &quot;best practice&quot;.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recommendation 18:</td>
<td>Provide support for subcontract management activities in the prime contract.</td>
</tr>
<tr>
<td>Recommendation 19:</td>
<td>A voluntary “Industry Code of Conduct” should be developed in accordance with the Trade Practices Act, 1974 Section 51AC to ensure a Systems Integrator or Outsourcing organisation discloses to the customer the subcontract management activities to be undertaken.</td>
</tr>
<tr>
<td>Recommendation 20:</td>
<td>Consider separate contracts - the first for defining the requirements and other documentation as necessary and the second for implementing the software.</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Recommendation 21:</td>
<td>As a strategy for avoiding possible action under the <em>Trade Practices Act, 1974</em>, predicted computer system performance estimates should not be provided by computer system developers unless they are accurate and correct, or unless heavily caveated so as to cause absolutely no misunderstanding as to their accuracy or correctness or the basis of their derivation.</td>
</tr>
<tr>
<td>Recommendation 22:</td>
<td><em>Performance</em> requirements should be explicitly stated in a contract (possibly by reference to an engineering specification) in order to have a contractually binding effect.</td>
</tr>
<tr>
<td>Recommendation 23:</td>
<td>As a strategy for avoiding possible claims under the <em>Trade Practices Act, 1974</em>, if a buyer requests that certain performance criteria of a system be met, then a supplier should <em>expressly</em> deny that it can meet these criteria if there is any doubt.</td>
</tr>
</tbody>
</table>

### Case Study 5 - Software Quality Assurance (See Section: 4.9)

<table>
<thead>
<tr>
<th>Recommendation 24:</th>
<th>Link the contract to Software Engineering “best practice”.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recommendation 25:</td>
<td>Prove support in the contract for software quality assurance activities.</td>
</tr>
<tr>
<td>Recommendation 26:</td>
<td>A voluntary “Industry Code of Conduct” should be developed in accordance with the <em>Trade Practices Act, 1974</em> Section 51AC to ensure a Systems Integrator or Outsourcing organisation discloses to the customer the Quality Assurance activities to be undertaken.</td>
</tr>
<tr>
<td>Recommendation 27:</td>
<td>Do not blindly call up specifications in the contract without first understanding the impact. Recognise that standards are generally written as general documents that may require some level of tailoring to match with the specific project aims.</td>
</tr>
</tbody>
</table>
| Recommendation 28: | Information Technology practitioners should be made aware that silence itself may constitute a breach of Section 52 of the *Trade Practices Act, 1974* as conduct which is misleading or deceptive (or at least conduct which was likely to mislead or deceive).  
If a group or team is formed consisting of representatives from different contracted parties, it is important to be active in the decision making process and in particular, raise clear objections to group decisions where appropriate. |

### Case Study 6 - Configuration Management (See Section: 4.10)
<table>
<thead>
<tr>
<th>Recommendation 29:</th>
<th>Link the contract to Software Engineering “best practice”.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recommendation 30:</td>
<td>Provide support in the contract for an effective Configuration Management process.</td>
</tr>
<tr>
<td>Recommendation 31:</td>
<td>A voluntary “Industry Code of Conduct” should be developed in accordance with the Trade Practices Act, 1974 Section 51AC to ensure a Systems Integrator or Outsourcing organisation discloses to the customer the Configuration Management activities to be undertaken.</td>
</tr>
<tr>
<td>Recommendation 32:</td>
<td>Detailed records should be maintained within an organisation to show actual costs of work. Such records may be used as evidence to help in the assessment of quantum.</td>
</tr>
<tr>
<td>Recommendation 33:</td>
<td>Detailed records should be maintained within an organisation of interactions between parties, including micro-agreements. Such records may be used as evidence to help in the assessment of liability.</td>
</tr>
</tbody>
</table>

### 4.13.3 Consolidated Recommendations

The recommendations in the above table are intended to give a broad coverage of the activities which, in the opinion of the author, should be conducted to minimise the risks and maximise the benefits associated with micro-agreements. To some extent the recommendations provided above can be condensed into a core set as follows.

- Link the contract to Software Engineering “best practice”.
- A voluntary “Industry Code of Conduct” should be developed in accordance with the *Trade Practices Act, 1974*, Section 51AC to ensure a Systems Integrator or Outsourcing organisation discloses to the customer the extent to which the following activities are intended to be undertaken: Requirements Management, Project Planning, Project Tracking and Oversight, Subcontractor Management, Quality Assurance, and Configuration Management.
- Support should be provided in a systems integration or outsourcing contract for at least the following activities: Requirements Management, Project Planning, Project Tracking and Oversight, Subcontractor Management, Quality Assurance, and Configuration Management.

Each of these recommendations has been discussed in detail above however it is the opinion of the author that these points summarise the critical aspects of software engineering “best practice” and its application to information technology systems integration and outsourcing contracts. By giving these elements an appropriate degree of contractual weight it is considered likely that the effect of micro-agreements will be to make the products being developed better overall, and to make the schedule, cost and quality of the product more controllable.
4.14 CONCLUSION

In conclusion, a "micro-agreement" means:

"an agreement or common understanding between two or more parties involved in an information technology or related development or acquisition project, which relates to some aspect of work associated with that project, but is not necessarily expressly or by implication covered by any existing or planned contractual clauses".

Micro-agreements may take many forms however it is important to note that the micro-agreement itself appears to constitute a special kind of interaction between the parties which is not necessarily formalised to any great extent, but nonetheless may have a significant effect on the outcome of an information technology systems integration or outsourcing project.

The case studies have shown that the legal effect of micro-agreements can be significant. Despite the fact that some micro-agreements may have the appearance being innocent informal day-to-day interactions between the parties involved in systems integration work, the effect can be significant in a number of ways.

From an engineering or project management perspective micro-agreements may result in adverse consequences such as increased cost and schedule and possibly lesser system capability than originally expected.

From a legal perspective their force comes from a range of fundamental legal devices including contract law, Trade Practices Act, estoppel, unconscionability, misrepresentation and many others as illustrated in this chapter.

Despite these pitfalls however, micro-agreements play an important role in any software development because it is only through continuous effective communication between the various parties involved that new issues which invariably become known will be efficiently resolved. It is considered that the recommendations made in this chapter, when implemented appropriately for each software development or systems integration task, are likely to provide a good balance between providing sufficient flexibility to accommodate change (including new ideas generally), while at the same time providing an appropriate degree of stability and control over cost, schedule and quality.
5. "Micro-Agreements"
Part 2: Relationship Models

The objective of this chapter is:
- to examine some of the legal relationships which may be present in larger information technology developments and to assess the impact of micro-agreements on each of these relationship models
- to "test" the theory of micro-agreements established in the preceding chapters to show that the principles are consistent across a number of different legal relationships models

5.1 INTRODUCTION

5.1.1 The Aim of this Chapter

The previous chapter introduced the new legal concept of a "micro-agreement" and identified how micro-agreements are likely to exist in information technology systems integration and outsourcing contracts. As discussed in that chapter, micro-agreements are fundamentally about communication. In any development with a large number of parties involved, effective communication is essential to ensure that all the components of a system can be successfully tied together. Micro-agreements play a part in this and to some extent include the interpersonal interactions which make up the totality of communication associated with a project.

The previous chapter focused on the micro-agreements themselves, providing a definition and case studies, identifying problems and benefits of them, and making recommendations for minimising the risks and maximising the advantages associated with them.

The aim of this chapter is to examine some of the different relationships which may exist between the parties to a information technology systems integration or outsourcing project, and to determine the potential impact of micro-agreements on the developments underway, while focusing on whether the type of legal relationship has an effect on the outcome despite the micro-agreement. Also of consideration is whether particular relationship models have a tendency to generate micro-agreements.

In order to achieve this aim the following approach is taken. Firstly a brief description is given of some of the types of relationship which can exist - not only legal relationships such as partnership, joint venture etc., but also some engineering or business relationships including "joint application development", "partnering" etc. The description of
these relationships is provided as a guide to the different types of basic relationship which may be formed between the parties, but is not intended to provide an exhaustive analysis of each topic.

In a later part of this chapter an analysis of some of the more specific legal relationship models used in systems integration and outsourcing projects will then be undertaken from the point of view of examining the impact of micro-agreements on development where particular relationship models are used. Some of the relationship models examined include: traditional multi-party contractual relationships, the formation of separate companies for the purpose of focusing on common objectives, joint venture arrangements, and service contract types of arrangements. The impact of micro-agreements on these relationship models is then assessed with the potential benefits and deficiencies identified and recommendations made.

Like the previous chapter, this one is also fundamentally about communication. Earlier on, Chapter 2 The Nature of Large Information Technology Projects - described some of the problems associated with large software developments including (among other things) the dynamic nature of how systems change throughout their life, the expected number of defects, and the difficulty in accurately estimating cost and schedule. These technical issues are reasonably well known and have been documented in the information technology literature for a considerable period of time\textsuperscript{504}. By no means however have all the problems been solved yet although some engineering solutions are certainly available for particular problems. As an example, the use of "object oriented\textsuperscript{505}" software development techniques may help to minimise the impact of requirements changes on the software architecture.

Similarly, attempts have been made from a legal perspective to minimise the adverse effects of the problems and to maximise the chance of success of the project. The traditional mechanism from a legal perspective is to ensure that an appropriate contract is in place which clearly defines the rights and obligations of all the parties. A number of computer contract precedent sets have been published in the literature\textsuperscript{506}.

In addition to these methods, an appropriate legal relationship can also be critical to the success of systems integration and outsourcing projects. By definition, outsourcing requires some form of relationship to be established with external organisations, and in systems integration work, it is often necessary to integrate different software and hardware components from separate suppliers, thus necessitating some form of relationship to exist between the parties involved.

Thus, we have seen from the previous chapter the effect of micro-agreements on such projects by examining particular micro-agreements in the context of specific factual

\textsuperscript{504} see for example the classic text: Brooks, Federick P. Jr. The Mythical Man-Month, 1975, Addison-Wesley
\textsuperscript{505} this is a way of structuring a system into components called "objects" and "classes" of objects where each object has particular "attributes" and operations associated with it. For example, a "bank account" object may have attributes such as "owner" and "account balance".
situations. This chapter examines the impact of micro-agreements on a number of different relationship models.

5.2 LEGAL RELATIONSHIPS

5.2.1 Introduction
The intention of this Section is to provide a basic outline of some of the different types of legal relationships which can exist between the parties to an information technology systems integration or outsourcing project. It is not intended in this Section to provide an exhaustive coverage of each sub-topic but rather, give a basic rundown of the principles, advantages and disadvantages of each of the following areas:

- Partnerships
- Joint Ventures
- Trusts
- Corporations
- Agency

Contractual relationships are paramount to any large information technology development and accordingly reference should be made to Chapter 3 for a detailed discussion of the legal issues involved in that regard.

Later on in this chapter, further discussion of particular relationships is provided in the context of common relationship models used in the industry.

5.2.2 Partnership

5.2.2.1 Definition
In Australia, a partnership is defined under the various state Partnership Acts as follows:\^{507}

"(1) Partnership is the relation which subsists between persons carrying on a business in common with a view of profit.

(2) But the relation between members of any company or association which is -

(a) registered as a company under any Act for the time being in force and relating to the registration constitution or incorporation of companies; or

(b) formed or incorporated by or in pursuance of any Act or letters patent or

Royal Charter -

is not a partnership within the meaning of this Act."

Apart from this statutory definition, the Partnership Act also supports the case law to the extent that "...rules of equity and of common law applicable to partnership shall continue in force except so far as they are inconsistent with the express provisions of the Act"\^{508}.

5.2.2.2 Basic Principles

---

\^{507} Partnership Act (NSW) Section 1, Partnership Act (Victoria) Section 5, Partnership Act (SA) Section 1, Partnership Act (Queensland) Section 5, Partnership Act (WA) Section 7, Partnership Act (Tasmania) Section 5, and Partnership Act (ACT) Section 6.

\^{508} Partnership Act (NSW), Section 46, Partnership Act (Vic), Section 4, Partnership Act (SA), Section 46, Partnership Act (Qld), Section 48, Partnership Act (WA), Section 6, Partnership Act (Tas), Section 5, Partnership Act (ACT), Section 5.
A partnership is not a separate legal entity but does establish a legal relationship between partners and accordingly imposes certain legal rights and obligations. Based on the statutory definition given in the Partnership Act, a number of elements are required to determine whether a partnership exists and these are discussed briefly below.

...the relation...
There appears to be no formal contractual or other requirement under the Partnership Act which is necessary to meet this element of the partnership definition, provided that there is some sort of agreement or understanding between the partners for the carrying on of a business in common with a view to profit.

Some restrictions do however exist such as a limit to the number of partners\textsuperscript{509}, and under some circumstances a partnership agreement which is for a period of greater than one year must be in writing in order to meet Statute of Frauds requirements for enforceability\textsuperscript{510}.

...carrying on a business...
The Partnership Act defines a business as including: "every trade, occupation or profession"\textsuperscript{511}. The wide scope of this definition obviously includes businesses associated with information technology systems integration and outsourcing. One question which arises however is, despite work being conducted within the scope of a "trade, occupation or profession", what level of continuity of the relationship is required to constitute a 'business'? As an example, could (say) a single transaction be the basis for finding that a partnership exists?

Simply put, case law indicates that a "business" involves a certain degree of continuity. In United Dominion Corporation v Brian Pty Ltd\textsuperscript{512} Dawson J. stated that:

"...whether one or two transactions make a business depends on the circumstances of each case. I take the test to be this: if an isolated transaction, if repeated would be a transaction in a business, is proved to have been undertaken with the intention that it should be the first of several transactions, that is, with the intent of carrying on a business, then it is a first transaction in an existing business. The business exists from the time of the commencement of that transaction with the intent that it should be one of a series."

Looking at the specific case of large information technology systems integration contracts, the problem with the above is that it may not be the intention of the parties to have a continuous business relationship, but rather to organise resources of a number of organisations for the purpose of developing a single information technology product (for example a banking system, communication system or military system). This could be the case despite the fact that the 'single' transaction is a relatively large development spanning months or even years. In such a case the question may arise as to whether a continuous business relationship (and a partnership) exists, or whether there is a single business transaction (which due to the size of the development may extend over a considerable period of time). This issue is discussed further in the Section below on joint ventures.

\textsuperscript{509} Corporations Law Section 112(2) limits partnerships to 20 partners except in the case of certain professions more than 20 partners are allowed: Corporations Law Section 111A.

\textsuperscript{510} Latimer, Paul: 1998 Australian Business Law, CCH, 1997 at p. 717

\textsuperscript{511} Partnership Act (NSW), Section 45, Partnership Act (Vic), Section 3, Partnership Act (SA), Section 3, Partnership Act (Qld), Section 3, Partnership Act (WA), Section 3, Partnership Act (Tas), Section 4, Partnership Act (ACT), Section 4.

\textsuperscript{512} United Dominion Corporation Ltd. v Brian Pty Ltd (1985) 157 CLR 1, per Dawson J at 15,
The main point to note however is that the relationship between parties to a information technology development needs to be examined carefully to determine whether they are conducting a ‘business’ within the meaning of the Partnership Act.\textsuperscript{513}

\textit{...in common...}
Under the partnership laws, an important issue is that for a partnership to exist the business needs to be carried out on behalf of all the partners with a common purpose. This is despite there being no requirement for all partners to be actively involved in the business. Agency is also an important issue and this aspect is discussed in a following Section.

\textit{...with a view of profit...}
The statutory definition clearly requires that the partners act in common with a view to profit. This provision appears to restate the common law dating back to cases such as \textit{Pooley v Driver}\textsuperscript{514} in which it was found that:

"There could not be a partnership without there was a commercial business to be carried on with a view to profit and for the division of profits."

The main issues associated with this requirement are firstly, non-profit organisations are unlikely to constitute partnerships\textsuperscript{515} (although that in itself is unlikely to be an issue in the information technology contracting field); and secondly, there is some provision in the \textit{Partnership Act} allowing for sharing of profits between partners\textsuperscript{516}.

\subsection{5.2.2.3 Liability}
The issue of liability is important to a partnership because the acts or omissions of some partners may, under some circumstances, result in liability for the others. The basis of liability of partners is different for contract and tort based actions against the partnership.

In contract based actions, partnership liability is joint - that is, all members of the partnership are jointly liable for any contractual obligations of the firm. Specifically, the \textit{Partnership Act} provides that:

"...Every partner in a firm is liable jointly with the other partners for all debts and obligations of the firm incurred while he is a partner, and after his death his estate is also severably liable in due course of administration for such debts and obligations so far as they remain unsatisfied but subject to the prior payment of his separate debts.\textsuperscript{517}"

Note however that some legislation relating to court operation has modified this restriction to allow joint and several actions against partners under some circumstances\textsuperscript{518}.

In tort, the \textit{Partnership Act} provides that the partnership is liable for the tortious acts and omissions of a partner.

\textsuperscript{513} Some other cases which have examined the issue whether a ‘business’ existed include: \textit{Smith v Anderson} (1880) 15 Ch D 247, \textit{French v Stiring} (1857) 140 ER 455, \textit{Tumbull v Ah Mow} (1871) 2 Australian Jurist Reports 40, and \textit{Checker Taxicab Co Ltd v Stone} [1930] NZLR 169.

\textsuperscript{514} \textit{Pooley v Driver} (1876) 5 Ch D. 458 at 472 per Jessel M.R.

\textsuperscript{515} see for example: \textit{Wise v Perpetual Trustee Co. Ltd}. [1903] AC 139 at 149; \textit{Bradley Egg Farm v Clifford} [1943] 2 All ER 378; \textit{Bohemians Club v Acting Federal Commissioner of Taxation} (1918) 24 CLR 334 at 337; \textit{R v Federal Court of Australia, ex parte Western Australian National Football League Inc.} [1979] 143 CLR 190

\textsuperscript{516} \textit{Partnership Act} (NSW), Section 24, \textit{Partnership Act} (Vic), Section 28, \textit{Partnership Act} (SA), Section 24, \textit{Partnership Act} (Qld), Section 27, \textit{Partnership Act} (WA), Section 34, \textit{Partnership Act} (Tas), Section 29, \textit{Partnership Act} (ACT), Section 29.

\textsuperscript{517} \textit{Partnership Act} (NSW), Section 9, \textit{Partnership Act} (Vic), Section 13, \textit{Partnership Act} (SA), Section 9, \textit{Partnership Act} (Qld), Section 12, \textit{Partnership Act} (WA), Section 16, \textit{Partnership Act} (Tas), Section 14, \textit{Partnership Act} (ACT), Section 13.

\textsuperscript{518} eg \textit{Supreme Court Act, 1970 (NSW)} Section 97
"...Where by any wrongful act or omission of any partner acting in the ordinary course of the business of the firm or with the authority of his co-partners loss or injury is caused to any person not being a partner in the firm or any penalty is incurred the firm is liable therefor to the same extent as the partner so acting or omitting to act. 519a

In addition, partners are also liable for the tortious acts of other partners:

"...Every partner is liable jointly with his co-partners and also severally for everything which the firm while he is a partner therein becomes liable under either of the last two preceding Sections. 520a

In summary, partnerships are a common way of establishing and defining the legal relationship between parties and as such a significant body of case law exists which is related to the liability of partners and partnerships 521.

5.2.2.4 Partnership and Agency

Partnership has been described as a branch of agency law although each partner is essentially an agent and a principle of the other partners at the same time:

"In partnership, there is a mutuality of rights and obligations. Each person is agent and principal of the other. Each can bind the other and can be bound by the actions of the other. In an agency, there is a one-way relationship, with one person the principal, and one the agent. The principal does not owe a fiduciary duty to the agent as does one partner to another. 522a

As the law in relation to agency has the effect of providing certain legal rights and obligations on the principle as a result of the actions of the agent, the issue of agency can be particularly important in larger information technology projects where many parties are involved. Where, for example, two corporations enter into a partnership arrangement to develop a computer system, it is conceivable that the actions of an individual employee of one corporation (one partner) may bind the other corporation (the other partner).

As micro-agreements can be an important part of any larger development, consideration of potential issues related to agency law need to be considered when forming an appropriate relationship model.

5.2.2.5 Advantages

Partnerships have a number of advantages as described below 522.

- Simplicity. A formal document is usually not required (although is advisable)
- Ease of change. As the rights and obligations of partners may be contract based, business arrangements can be relatively easily altered by modifying the partnership agreement.

519 Partnership Act (NSW), Section 10, Partnership Act (Vic), Section 14, Partnership Act (SA), Section 10, Partnership Act (Qld), Section 13, Partnership Act (WA), Section 17, Partnership Act (Tas), Section 15, Partnership Act (ACT), Section 14.
520 Partnership Act (NSW), Section 12, Partnership Act (Vic), Section 16, Partnership Act (SA), Section 12, Partnership Act (Qld), Section 15, Partnership Act (WA), Section 19, Partnership Act (Tas), Section 17, Partnership Act (ACT), Section 16.
521 for example, see: National Commercial Banking Corporation of Australia v Batty (1986) 160 CLR 251; Walker v European Electronics Pty Ltd (in liq) (1990) 23 NSWLR 1; Petrou v Hatzigeorgiou (1991) Aust Torts Reports ¶81-071; Polkinghome v Holland (1934) 51 CLR 143.

Paul P. Parnell

188
• Secrecy. Registration of partnership documents is generally not required thus giving a greater degree of privacy than for example a company.

• Tax advantages. A partnership itself is not a legal entity and does not pay tax (although the partners do). Taxation advantages may include: income splitting can be made between partners in a preferred distribution; negative gearing can be used to minimise tax where a partner has other income; and capital gains tax advantages may be achievable through averaging or indexation.

5.2.2.6 Disadvantages

Despite the advantages listed above, there are also disadvantages with a partnership:

• Liability. There is no limit on liability of a partnership although a "limited partnership" will provide protection for some of the partners. The liability of partners in a normal partnership is effectively joint and several (as discussed previously).

• Agency. Actions of one partner may bind the other partners.

• Size of Firm. The number of partners is limited.

• Trading of an interest in a partnership is not as easy as, for example, shares. Partnership interests are not traded on the stock exchange, but rather are sold through private advertising etc.

• Management of the partnership may at times be difficult, particularly as certain decisions may require a unanimous decision of all the partners.

5.2.3 Joint Venture

5.2.3.1 Definition

A "joint venture" has been described in the following terms:

"A joint venture is an association of persons, natural or corporate, who agree by contract to engage in some common, usually ad hoc undertaking for joint profit by combining their respective resources, without, however, forming a partnership in the legal sense (of creating that status) or corporation; their agreement also provides for a community of interest among the joint venturers each of whom is both principal and agent as to the others within the scope of the venture over which each venturer exercises some degree of control."

In the Section above on Partnership the issue was raised, in relation to larger information technology systems integration projects, as to whether a partnership could exist when it was the intention of the parties to combine resources for the purpose of developing a single large system, but not to have a continuing business relationship beyond that. Although a partnership generally requires a continuing business relationship, there is scope for interpreting what exactly this means. Commentators such as Webb for example have stated that:

"... it is perfectly possible to have a partnership in respect of one particular deal, transaction, or adventure."

In arriving to the above conclusion, Webb gives the example where a syndicate purchasing a block of units in a one off transaction, may in fact be found to be a partnership if they develop, improve, subdivide or otherwise deal with the land, "they will

---


run the risk of being found to be in a partnership if their activities are intended to produce profit for themselves - for they would then be carrying on a ‘business’. 526

Despite this, Australian Courts have generally not found a partnership to exist where there has been a traditional joint venture arrangement 527.

5.2.3.2 Principles

Some important points about joint ventures are discussed below.

Fiduciary Relationship

Even if no formal partnership or joint venture agreement is in place, it is possible that a fiduciary relationship exists between prospective partners or joint venturers. Rights and obligations may thus arise in the early stages of a joint venture or partnership which the parties should be aware. Specifically,

“A fiduciary relationship can arise and fiduciary duties can exist between parties who have not reached, and who may never reach, agreement upon the consensual terms which are to govern the arrangement between them. In particular, a fiduciary relationship with attendant fiduciary obligations may, and ordinarily will, exist between prospective partners who have embarked upon the conduct of the partnership business or venture before the precise terms of any partnership agreement have been settled. Indeed, in such circumstances, the mutual confidence and trust which underlie most consensual fiduciary relationships are likely to be more readily apparent than in the case where mutual rights and obligations have been expressly defined in some formal agreement. Likewise, the relationship between prospective partners or participants in a proposed partnership to carry out a single joint undertaking or endeavour will ordinarily be fiduciary if the prospective partners have reached an informal arrangement to assume such a relationship and have proceeded to take steps involved in its establishment or implementation. 528

A joint venture may also be a partnership

If a joint venture arrangement meets the criteria of a partnership then it will naturally constitute a partnership. A “partnership is the relation which subsists between persons carrying on a business in common with a view of profit.” 529 If a joint venture meets this criteria then it may be a partnership whether it is intended to be or not.

Liability

Like partnerships, the liability of members for debts of a joint venture is generally joint and several. It can be varied however by agreement between the partners 530.

5.2.3.3 Advantages

Some of the advantages of a joint venture are that:

- There is a degree of independence of members of a joint venture where “The joint venturer is not responsible for the acts of co-venturers” 531

526 ibid
528 United Dominions Corporation Ltd. v Brian Pty Ltd (1985) 59 ALJR 676; Despite the above, later judgements have introduced some uncertainty into the issue of fiduciary relationship of joint venturers: Australian Oil & Gas Corporation Limited v Bridge Oil Limited, NSW Court of Appeal, Kirby P, 12 April 1989, unreported.
529 Partnership Act (NSW) Section 1, Partnership Act (Victoria) Section 5, Partnership Act (SA) Section 1, Partnership Act (Queensland) Section 5, Partnership Act (WA) Section 7, Partnership Act (Tasmania) Section 6, and Partnership Act (ACT) Section 6.
• Taxation. Under the *Income Tax Assessment Act, 1936*, income and expenditure under a joint venture is not shared between members collectively, but rather, brought into account individually. "This allows each joint venturer to determine its own method of treating expenditure, valuing shares and, in particular, its own attitude as to what elections should be made under the Income Tax Assessment Act 1936." 532

• Disposing of assets can be an individual decision of each joint venturer and is not subject to, for example, the rules for disposal of assets of a partnership (unless the joint venture is also a partnership). Following on from this, "The transfer or assignment of rights of a party to a joint venture can take place without consent." 533

• Under some circumstances joint venturers may compete with each other. 534 Care must be taken however where a fiduciary relationship exists not to breach such obligations.

5.2.4 Trusts

5.2.4.1 Definition and Types of Trust
There are a number of different categories of trusts which broadly speaking, can be classified as either express or non-express trusts. Other classifications also exist but in general terms:

"A trust may be defined as obligation enforceable in equity which rests on a person (the trustee) as owner of some specific property (the trust property) to deal with that property for the benefit of another person (the beneficiary) or for the advancement of certain purposes." 535

Trusts can have a variety of characteristics and the following diagram is intended to provide a basic 'map' of the different types of trust which can exist. 536

533 Ibid
534 Ibid
It should be noted that the above figure shows one particular way of categorising trusts although because of the range and diversity of the different types of trusts which exist, other ways of categorising them also exist. The classification of different types of trust into a particular structure has not been absolutely defined from a legal perspective. It has been said that the different categories within which trusts fit are therefore not "sharply defined...[and in fact] there is much to be said for the view that classification is often unnecessary and tends to distract attention from more important questions."

Despite this, categorising the different types of trusts in a way such as shown in the diagram above allows for a somewhat more orderly and concise description suitable for the present purposes.

5.2.4.2 Express Trusts

An express trust is created where there is a clearly expressed intention, provided that certain other legal criteria are established. An express trust does not need to be in writing although it does need certainty in relation to a number of matters. Firstly, an express trust requires certainty of intention to create the trust; secondly, certainty of the subject matter

of the trust; and thirdly, certainty of the object of the trust. It is only when these three elements are satisfied that a valid trust is created.

Even in the case of discretionary trusts, there must still be a degree of certainty for the trust to be valid. Although it is true that when a discretionary trust comes into existence the trustee has certain discretionary powers in relation to the distribution of trust property to the beneficiaries, to allow for some order and certainty in the application of the law in relation to discretionary trusts, certain legal rules have been established to allow for the identification of specific beneficiaries and trust objects. Specifically, the object of the trust must satisfy the "criterion certainty" rule, that is, it must be possible for a particular individual to be identified as either belonging to a class or range of objects or not: McPhail v Doulton.

A fixed express trust, on the other hand, must identify with sufficient clarity the objects of the trust from the outset. It does not provide scope for discretionary dealings with the trust property but has explicit objects identified in accordance with the "list certainty" rule.

A bare trust (sometimes called a simple trust) is one where the trustee merely holds the trust property and has no active duties apart from distributing the trust property as directed by the beneficial owner. A special trust, on the other hand, covers all other cases where the trustee has active duties to perform.

A constituted trust is one where the property which is the subject of the trust is completely vested in the trustee as legal owner. Where some or all of the property is not completely vested in the trustee as legal owner, a non-constituted trust is established.

An executed trust is one where the intentions of the settlor are completely detailed in the trust documentation. An executory trust is one where further documentation or acts are needed to fully define the intentions of the settlor. Accordingly, in an executed trust the fiduciary obligations can be determined from the trust documentation itself, whereas for an executory trust, the full fiduciary obligations need further acts or instruments to be fully defined.

In addition to the above categories, trusts can also be classified as private or public trusts. In the case of a private express trust for example, a trust instrument, usually in the form of a deed, may be established which sets out (among other things) the rights and obligations of the parties, the beneficiaries, trust property and trust objects.

5.2.4.3 Non-Express Trusts

From the point of view of information technology projects, (and particularly larger ones) it is envisaged that any express trust would come into existence through a formally documented arrangement such as a trust deed. That instrument ideally would identify all relevant aspects of the trust including trust objects, beneficiaries, trust property, and the rights and obligations of the parties. As we have see however from the previous chapter, situations may exist where trusts can come into existence without the express intention of the parties. Case Study 3 (Section 4.7 above) provided a discussion and example of a situation where a constructive trust may exist in the context of an information technology development and the main issues associated with such a trust are outlined below.

5.2.4.3.1 Constructive trusts

McPhail v Doulton [1971] AC 424
Evans, M. Outline of Equity and Trusts, Butterworths, 1998, p.182
Where there is no explicit trust in place then under appropriate circumstances equity may intervene and create a constructive trust. "A constructive trust is a trust imposed by operation of law, regardless of the intentions of the parties concerned, whenever equity considers it unconscionable for the party holding title to the property in question to deny the interest claimed by another."\(^{542}\)

In applying constructive trusts, Deane J. in Muschinski v Dodds\(^{543}\) stated however that limits should be applied to the application of constructive trusts. He said that the constructive trust cannot be used as "a medium for the indulgence of idiosyncratic notions of fairness and justice"; and also that it can be used only when "warranted by established equitable principles or by legitimate processes of legal reasoning, by analogy, induction and deduction, from the starting point of a proper understanding of the conceptual foundation of such principles."\(^{544}\)

Despite this, Deane was prepared to find that a constructive trust did exist in the Hospital Products v USSC\(^{545}\) case despite the fact that no fiduciary duty was present. Specifically, in that case Deane J. stated (obiter):

"... USSC was entitled to a declaration that HPI was liable to account as a constructive trustee for the profits of that Australian business in accordance with the principles under which a constructive trust may be imposed as the appropriate form of equitable relief in circumstances where a person could not in good conscience retain for himself a benefit, or the proceeds of a benefit, which he has appropriated to himself in breach of his contractual or other legal or equitable obligations to another."

Overall, "a decree of constructive trust can be made where the defendant has been enriched at the plaintiff's expense and where, in the circumstances, it would be unjust for him to retain that benefit."\(^{546}\)

5.2.4.3.2 Implied, Resulting, or Presumptive trusts

As the name suggests, implied trusts are those where the law presumes that the requisite intention to create a trust exists, but where that intention may not be expressed by the creator. The names "resulting trust" and "implied trust" can be used interchangeably.\(^{547}\)

There are generally two ways in which an implied trust can come into existence. The first is where there is an incomplete disposition of property, through either a surplus of property after fulfillment of the original purpose; or where an express trust cannot be carried out for some reason (such as the death of the beneficiary).

As an example, in Smith v Cooke [1891] AC 297, a firm that became insolvent was conveyed on trust to a trustee for the purpose of using the business assets and income to repay creditors. After repayment of the amounts owed, there was a surplus. The court held that under the trust deed it was expressed that there was to be an absolute assignment of all business assets to the creditors in return for a release from liability for the debts. The surplus business assets were thus found to come under an implied trust to be held for the benefit of the creditors.

\(^{542}\) Evans, M. Outline of Equity and Trusts, Butterworths, 1998, page 239
\(^{543}\) Muschinski v Dodds (1985) 160 CLR 583, Deane J.
\(^{544}\) Muschinski v Dodds (1985) 160 CLR 583, Deane J. at 616
\(^{545}\) Hospital Products v United States Surgical Corporation (1984) 156 CLR 41
\(^{546}\) Evans, M. Outline of Equity and Trusts, Butterworths, 1998, page 251
\(^{547}\) Evans, M. Outline of Equity and Trusts, Butterworths, 1998, page 230
The second way in which an implied trust may be found to exist is where property is purchased in another person's name. Simply put, where a person buys property in another person's name and becomes the legal owner, then there will be an implied trust to the extent that the legal owner holds that property on trust for the for the person who pays the money for the property. In *Dyer v Dyer* it was held that:

"The clear result of all the cases, without a single exception, is, that the trust of a legal estate, whether freehold, copyhold, or leasehold; whether taken in the names of the purchasers and others jointly, or in the name of others without that of the purchaser; whether in one name or several; whether joint or successive, results to the man who advances the purchase-money. This is a general proposition supported by all the cases, and there is nothing to contradict it; and it goes on a strict analogy to the rule of the common law, and that where a feoffment is made without consideration, the use results to the feoffor. It is the established doctrine of a Court of equity, that this resulting-trust may be rebutted by circumstances in evidence."

Despite this the intention to create a trust may be rebutted with the appropriate evidence.

5.2.4.4 Fiduciary Obligations

A trustee operates under a fiduciary obligation to act in the interest of the beneficiaries ahead of his own interest. Equity imposes fiduciary obligations on a number of relationships including some which are of particular interest in the context of this chapter. For example, certain business and commercial arrangements including Trustee⇒Beneficiary, Director⇒Company, Partner⇒Partner, and Agent⇒Principal are fiduciary in nature and thus a fiduciary duty is owed by one party to another. The categories of relationship also includes others such as solicitor⇒client.

Once a fiduciary relationship exists then the fiduciary obligations include avoiding conflict of interest situations and avoiding making a profit at the expense of the other party. The nature of the relationship was summed up in the *Hospital Products Ltd v United States Surgical Corporation*:

"The critical feature of these relationships is that the fiduciary undertakes or agrees to act for the benefit of the beneficiary. This fiduciary relationship would be breached if a trustee acted for private advantage of a person who was not a beneficiary and, as a result, the beneficiary became disadvantaged or suffered some detriment. A court of equity would not allow a person in a fiduciary position to make a personal profit (unless expressly entitled) or to be placed in a conflict of interest situation."

5.2.4.5 Advantages

- A corporate trustee can be appointed
- A unit trust are not taxable if it distributes its profits
- Can have hybrid/discretionary trusts

5.2.4.6 Disadvantages

- tax losses are trapped within the trust
- no capital gains tax rollover benefits
- fiduciary duties are owed by the trustees to the beneficiaries

---

548 *Dyer v Dyer* (1788) 2 Cox 92 at 93
549 See for example *Bloch v Bloch* (1981) 37 ALR 55
550 *Hospital Products Ltd v United States Surgical Corporation* (1984) 156 CLR 41 pp 96-97

Paul P. Parnell

195
5.3 ENGINEERING AND BUSINESS RELATIONSHIPS

Apart from the legal relationships described in the previous Sections, a number of engineering or business relationship structures are often used. Some of these types of relationships are described below.

5.3.1 Joint Application Development

5.3.1.1 Definition

A Joint Application Development (JAD) approach is one where the parties to a development work together to achieve a mutually desirable outcome. Specifically, various interested groups such as the developer, customer and management meet, usually together with a trained facilitator, for the purpose of identifying requirements of the system to be developed. As has been seen in previous chapters, the management of requirements can be one of the most critical aspects of an information technology development because errors in the identification and management of requirements can lead to significant cost overruns and schedule delays.

Joint Application Development was designed to overcome some of these problems by bringing together interested parties to collect requirements.

Some commentators have indicated that Joint Application Development is the best method for collecting requirements.\textsuperscript{551}

5.3.1.2 Principles

Thus Joint Application Development is a method where both the user and the developer (and other interested parties) work together to identify the requirements of the system. It is usual that a facilitator is also present to aid these parties in the process of reaching a consensus on the requirements to be implemented. This method was developed in Canada in the 1970's and is now used effectively by information systems developers. Overall, if implemented correctly, it has been estimated that this method may cut the rate of requirement change by around half throughout the life of the project.\textsuperscript{552}

5.3.1.3 Advantages

Some of the main advantages of Joint Application Development as follows:

- "Saves time, eliminates process delays and misunderstandings and improves system quality"\textsuperscript{553}
- Reduces function creep\textsuperscript{554} (function creep is a term used to describe a trend of increasing the functionality of the system during its development as more requirements become known and the system evolves).
- "Lays the foundation for a framework of mutual education, separate brainstorming, binding negotiation, and progress tracking"\textsuperscript{555}

Overall, it is likely that a Joint Application Development approach will aid in the requirements identification and provide a set of requirements which are neither too vague

\textsuperscript{553} Hollander, Nathan, Naomi Mirlocco: "Facilitating workshops: Empowering the User to Develop Quality Systems Faster", Industrial Engineering, Oct., 1993
\textsuperscript{554} Anthes, Gary: "No More Creeps", Computerworld, May 2, 1994; Strehlo, Kevin: "Catching up with the Joneses and 'Requirement Creep'; InfoWorld, 29 July 1996; and Whitmore, Sam: "Readers Shed Development Woes", PC Week, 29 May 1995
nor too specific. It is also likely that due to competing interests of members involved in the Joint Application Development, that there will be less "gold plating" and requirements creep associated with the product under development.

In Section 0: Case Study 1 - Micro-agreements in the Context of Requirements Management the importance of managing requirements was discussed and certain recommendations were made. Joint Application Development is generally recognised within the information technology industry as one way of enabling requirements to be identified early in the development cycle. It is recommended that Joint Application Development should be incorporated, where appropriate, in the development effort with the aim of producing a software specification which can then be used to guide the implementation of the system.

5.3.2 Partnering

5.3.2.1 Definition

"Partnering" is a management technique which traditionally has been used in the construction industry and in the larger government contracting arena to achieve a level of cooperation and teamwork between the parties. It allows the parties to put aside strict legal rights to the extent necessary to focus on important issues of mutual benefit. More specifically, it has been described in the following terms:

"Partnering is a structured management approach to facilitate teamworking across contractual boundaries. Its fundamental components are formalised mutual objectives, agreed problem resolution methods, and an active search for continuous measurable improvements. It should not be confused with other good project management practices, or with long standing relationships, negotiated contracts or preferred supplier arrangements, all of which lack the structure and objective measures that must support a partnering relationship."  

"Partnering is thus intended to "look beyond the strict bounds of the contract to develop [a] co-operative working relationship which promotes common goals and objectives."

Further details of the way in which Partnering works, together with some of the associated advantages and disadvantages are listed below.

5.3.2.2 Principles

As mentioned above, Partnering aims to give all parties an incentive to cooperate in the achievement of goals, even if not entirely consistent with strict contractual clauses. It aims to achieve this through establishing:

- formalised mutual objectives,
- agreed problem resolution methods, and
- an active search for continuous measurable improvements

The actual mechanism by which these aims are achieved may vary although a "partnering workshop" conducted with the aid of a facilitator is one common method. Usually the partnering workshop is held early after contract award and is attended by all interested parties, including the buyer, developer and end user. Overall, it aims to establish ground

rules and guidelines which are used by the parties to achieve mutual interests. Many issues associated with large developments tend to result from communication problems. Partnering is intended to aid communication.

“At the beginning of their contractual relationship, the parties establish communication channels designed to promote openness, trust and efficient contract administration”.559

5.3.2.3 Advantages
Overall, the main advantages of Partnering appear to be related to cost reduction, schedule slippage minimisation and quality improvement.

In the literature, claimed advantages of Partnering include that it:

“establishes mutual goals and objectives... builds trust and encourages open communication... helps eliminate surprises... enables the parties to anticipate and resolve problems... avoids disputes through informal conflict management procedures... avoids litigation through the use of Alternative Dispute Resolution... reduces paperwork... reduces the time and cost of performance... reduces administration and oversight... improves safety... improves engineering efforts... improves morale and promotes professionalism in the workforce... generates harmonious business relationships... and focuses on the mutual interests of the parties.560

The above claims appear to be extensive although in some cases appear to somewhat subjective and unsubstantiated. Despite this, some quantitative data is available:

“In a 1992 study comparing the performance of the 19 projects that the Corps had partnered to date with the performance of a sample of non-partnered projects of the same size, cost increases were reduced by 9.12 percent; schedule growth by 7.19 percent; change order costs by 7.15 percent; claims costs were down 4.19 percent; and value engineering savings were nearly 15 times as great.”561

Regardless of the above claims, it appears however that the partnering process itself: “is worthwhile even on problem projects because it does force issues out into the open and escalates them more quickly for resolution”.562

5.3.2.4 Disadvantages
A number of commentators have discussed the disadvantages of partnering and in summary:

• “Partnering is not a substitute for experienced, compatible and skilled personnel doing the job. If the mix of people and skills is wrong, all that partnering will do is surface the problem more quickly.”563


Paul P. Parnell
"Too much emphasis on avoiding disputes can create an "avoidance at any cost" mentality which is not productive; place the emphasis on resolution."  

One of the biggest dangers with partnering appears to be that, at least in the United States it is marketed as having no effect on the strict contractual rights of the parties but rather, is simply a management tool for achieving greater co-operation between to parties to achieve mutual gain. One example of this type of approach is illustrated in the following passage:

"Partnering is not a contractual agreement and does not create, relinquish, or conflict with the legally binding rights or duties of the parties."  

It is submitted that this statement is incorrect or at least misleading under Australian law. As we have seen in a previous chapter "Micro-Agreements" Part 1: Concepts and Case Studies", many situations can arise where, despite a contract being in place, the conduct of the parties will have the effect of modifying legal rights and obligations. Examples include issues involving estoppel, misleading and deceptive conduct, unconscionability, etc.

5.3.2.5 Conclusion
The primary aim of partnering is simply to achieve a better result for the parties involved, mainly through communication and dispute resolution mechanisms. The United States Army Acquisition Office summed up the method in the following way:

"Too often the acquisition process is undermined by adversarial relationships, suspicion between the [buyer] and the [seller], volumes of paperwork and costly litigation. We can no longer afford to do business in this manner."  

Overall, the basis of the partnering approach therefore is to provide an environment where rewards are gained through working together.

From the point of view of information technology contracts there is likely to be some scope for using a partnering approach, although in practice it is usually put in place after a contract has been awarded. We have seen in previous chapters that one of the fundamental issues associated with larger information technology developments is the amount of uncertainty and change throughout the life of the project. Accordingly, it is submitted that a better approach may be to formalise partnering by way of an appropriate legal relationship prior to the development so that maximum advantage can be taken of the law.

Following Sections in this chapter describe a number of relationship models which may provide some or all of the benefits of partnering (as described above), while at the same time giving legal force to the relationship through, for example, a partnership, joint venture or company structure.

The literature indicates that the use of joint ventures is probably less common in Australia (for large information technology projects) than some of the other legal relationships available such as partnerships or corporate structures\textsuperscript{568}.

The reason for this is likely to be due to a combination of factors such as balancing the cost and complexity of establishing and maintaining the respective legal relationships; against the benefits of achieving a relationship which is capable of coping with the changing business needs of the parties.

Partnering approaches and management techniques such as “integrated Product Teams” can relatively easily gain legal status through, for example, a partnership of companies which have a common aim such as the development of a large integrated computer system.

5.4 RELATIONSHIP MODELS

5.4.1 Introduction

In large information technology systems integration projects it is possible for the parties to enter into a variety of commercial arrangements to achieve their aims. This Section is intended to identify some of the models that have been used for systems integration contacts and establish, in general terms, the strengths and weaknesses of each model together with an assessment of the impact of micro-agreements on each of these.

It has already been discussed that one of the prime difficulties with information technology developments, and particularly larger systems integration and outsourcing projects is that of communication between the parties. In order to identify and resolve issues associated with such projects, certain recommendations were made (in Chapter 4) which included such things as ensuring that core activities are performed. These activities include: effective project planning, project tracking and oversight, requirements management, configuration management, quality assurance and subcontractor management.

One of the key points to note is that all these activities require effective communication between the parties. Accordingly, the relationship which exists between the parties may have a significant effect on the outcome. As a simple illustration, if a householder requires his lawn mowed and garden watered at the same time and contracted two workers to do these separate jobs, then some degree of co-operation between the two workers would be desirable in order to avoid either injury from the lawn mower, or a wet lawnmower ignition. Despite the fact that each worker may have a binding contract with the householder to perform the task within the contracted time, a strict reliance on legal obligations under the respective contracts would not necessarily result in the best outcome. Rather, a co-operative arrangement between the two workers such as a joint venture arrangement for garden care services may be more appropriate.

Although somewhat simplistic and obvious, this illustration is intended to show exactly how a significant number of information technology contracts are written — that is, in terms of identifying individual rights and obligations but not addressing a co-operative method of achieving those aims. The relationship which exists between the various parties is important and it is the intention of this chapter to discuss the advantages and

\textsuperscript{568} Pollard, S. “Managing Risk in Major IT Projects”, 1996, http://www.gtlaw.com.au/gt/bin/frameup.cgi/gt/pubs/manrisk.html it is stated that “the fourth model [joint venture change agent model] is unusual but will become increasingly popular in Australia”
disadvantages of a number of the different types of relationship models which can exist, particularly in the context of micro-agreements. Specifically, the following relationship models are discussed:

- Conventional Model
- Service Model
- Captive Model
- Joint Venture Change Agent Model

A description of each of these is provided in following Sections. In the context of micro-agreements, it is also intended to assess how likely they are to affect information technology projects and specifically, to identify whether any particular types of relationship models are more suitable than others to maximise the benefits and minimise the problems associated with micro-agreements.

5.4.2 Method of Analysis of Relationship Models

In the previous chapter it was asserted and recommended that certain fundamental software engineering practices be put in place in order to maximise the benefits of micro-agreements and minimise any adverse effects arising from them. In light of those assertions, an analysis of the different relationship models is conducted below with the aim of answering the fundamental questions:

What is the effect of a micro-agreement on a development or system integration project which is undertaken using a particular relationship model? In other words, how does each particular relationship model cope with micro-agreements, and in addition, do any relationship models have a greater tendency to generate micro-agreements?

The aim therefore is to conduct a structured analysis across a number of relationship models using common assumptions and constraints. A common basis for comparison is needed to make a relative assessment of the benefits and deficiencies of each relationship model and this basis is described below. The comparison of relationship models presented in this chapter builds on the discussion presented in Chapter 4: "Micro-Agreements" Part 1: Concepts and Case Studies, and uses the same fundamental software engineering practices as the starting point. These practices include Requirements Management, Software Project Planning, Software Tracking and Oversight, Sub-contractor Management, Software Quality Assurance and Software Configuration Management. In the previous chapter, examples of micro-agreements were provided in the context of each of these areas and it is the aim of this chapter to analyse each relationship model by addressing the same fundamental software engineering practices but in a different context.

In summary, the following diagram and table describe this approach to providing a framework for comparison. The answers to the evaluation criteria listed in the table are used to assess the effectiveness of each relationship model in coping with micro-agreements.
The following table provides the full list of evaluation criteria as shown in the diagram above. The answers to questions posed in the evaluation criteria are used to evaluate each relationship model's effectiveness in dealing with micro-agreements.

### Evaluation of Business and Technical Risk

<table>
<thead>
<tr>
<th>Software Engineering Best Practice Area</th>
<th>Relationship Model Evaluation Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Micro-agreements in the context of <strong>Requirements Management</strong></td>
<td>• Does the relationship model have a capacity to deal with inadequately defined requirements?</td>
</tr>
<tr>
<td>• Does the relationship model have a capacity to deal with an evolving system architecture?</td>
<td></td>
</tr>
<tr>
<td>• Does the relationship model have a capacity to deal with a system which needs to grow?</td>
<td></td>
</tr>
<tr>
<td>Micro-agreements in the context of <strong>Software Project Planning</strong></td>
<td>• Does the relationship model have the capacity to deal with developments where the schedule is important?</td>
</tr>
<tr>
<td>Micro-agreements in <strong>Evaluation Criteria</strong></td>
<td>• Are the overheads low for this relationship model?</td>
</tr>
<tr>
<td>Software Engineering Best Practice Area</td>
<td>Relationship Model Evaluation Criteria</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>----------------------------------------</td>
</tr>
</tbody>
</table>
| the context of Software Tracking and Oversight | • Does this relationship model give the customer good visibility of progress?  
• Does the relationship model give management good visibility of progress? |
| Micro-agreements in the context of Sub-contractor Management | • Does the relationship model allow the management of risks effectively? |
| Micro-agreements in the context of Software Quality Assurance | • Does the relationship model have the capacity to deal with developments where system reliability is important?  
• Does this relationship model require sophisticated developers and management? |
| Micro-agreements in the context of Software Configuration Management | • Does the relationship model have the capacity to enforce good software configuration management practices? |

**Evaluation of Legal Risks**

| Legal Risk | • What capacity does the particular relationship model have in minimising legal risks to the parties involved? Specifically, what is the likelihood of claims under contract, estoppel, trade practices or other legal doctrines if this relationship model is used? |

*Figure 29: Relationship Model Evaluation Criteria*
5.5 CONVENTIONAL RELATIONSHIP MODEL

5.5.1 Description
In this model the customer itself takes on the role of the systems integrator by calling in, or outsourcing, the expertise and products of various sub-contractors as required\textsuperscript{569}.

5.5.2 Example
If a customer required the development of a flight simulator for training crews, then it could possibly use the conventional contracting model to acquire the different sub-systems for integration. It may, for example need to integrate two different navigation sub-systems (for redundancy) such as a satellite based Global Positioning System (GPS) sensor together with an Inertial Navigation System. As these two sources of navigation data need to be integrated into the flight simulation model the types of data, data rates etc. need to be consistent. In this situation, the systems integrator could contract with the suppliers of each of the two navigation systems directly to provide the required hardware and software.

5.5.3 Analysing this Relationship Model
Based on the method of analysis discussed above in Section 5.4.2, the starting point for analysis of this model is to address the evaluation criteria provided in that Section.

5.5.3.1 Micro-agreements in the context of Requirements Management
The concept of "Requirements Management" was discussed in the previous chapter. In summary:

"Requirements Management involves establishing and maintaining an understanding and agreement with the customer on the requirements for the software throughout the software life cycle. The agreements cover both the technical requirements for the software and the non-technical requirements, such as delivery dates for the software. The agreements form the basis for estimating, planning, performing and tracking the project's software activities."\textsuperscript{570}

5.5.3.1.1 What capacity does this relationship model have in dealing with inadequately defined requirements?
The issue of micro-agreements that occur in the context of requirements management was discussed above in Section 0. In that case study a particular set of facts was

presented which showed how a micro-agreement resulted in a number of potential problems (see Section 4.5.3). In summary, the issues were related to the fact that despite requirements specifications being available at the start of the integration, during the course of development, some of those requirements changed (that is, some of the *interface* requirements in that particular case). As a result, the product was eventually delivered late, cost more, and was technically inferior to what could have been delivered if the particular micro-agreement was put in place.

The point to note is that, particularly in larger projects, requirements change. How well does the conventional contracting model cope with this? From a simplistic view a sub-contract would probably specify the equipment to be procured and would define all functional, performance and interface requirements. It would thus provide (in theory) a degree of certainty in what was to be delivered as well as certainty as to price, schedule and quality. Despite this it is likely that requirements will change during the life of the project. *Section 2.5 Requirements Management* and *Section 0 Case Study 1 - Micro-agreements in the Context of Requirements Management* provided a discussion on the amount of change and the reasons for change.\(^\text{571}\)

In order to overcome the risk and difficulties associated with changes or variations, invariably contracts provide a number of ways of coping with change. These include such things as variation clauses and dispute resolution mechanisms (such as compulsory mediation in the first instance) when a dispute arises. These types of mechanisms however tend towards fixing the problems after they occur rather than preventing them from occurring in the first place.

In other words, these and other similar mechanisms attempt to “patch up” what could be regarded as a fundamentally flawed contractual situation when applied to larger systems integration contracts. The “flaw” arises because on the one hand the parties want (and need) certainty of rights and obligations, while on the other hand history has shown that software development activities, and particularly those associated with the development of large systems, are likely to be dynamic in nature and require changes in requirements throughout the life of the system. In other words, how then can certainty be achieved when things are changing so much?

In order to cope with this situation some projects start with a separate contract for the production of a detailed requirements specification. Once this is available then a second contract is let for the implementation of the system. Although this is likely to help the situation by providing a good understanding of the requirements before the project commences (and is thus worthwhile considering), it does not overcome the problem totally because external factors may still cause the requirements to change later on. External factors influencing the situation can be such things as keeping up with a competitor’s product for example.

Good requirements management practices are also essential and these were discussed above in a reasonable amount of detail in *Section 4.5.4*. In that Section it was discussed how a solid software engineering process may help to avoid the problem to some extent.

Despite these and various other workarounds, it is submitted that the conventional model is probably best suited for straight-forward projects where, for example, a product already exists and needs to be integrated into a larger development. To be successful, the requirements need to be well defined and understood so that the amount of uncertainty present when entering into the contract is minimised.

In summary it is considered that the conventional model, using a single contract, may not be particularly good at coping with changing requirements because of the factors mentioned above.

Overall, for the reasons given above the:

Conventional model assessed as: Poor

for this criteria.

5.5.3.1.2 What capacity does this relationship model have in dealing with an evolving system architecture?

Before addressing this question, a brief set of definitions are in order. Remember in Chapter 2: The Nature of Large Information Technology Projects above where lifecycle models were discussed? The discussion there listed the basic steps associated with the development of a information technology system and in summary (stated simply), these were:

| Phase 1 - Requirements Analysis | i.e.: Identify what the system has to do. |
| Phase 2 - Design | i.e.: Define how the system is to do it. |
| Phase 3 - Implementation | i.e.: Build the system. |
| Phase 4 - Verification | i.e.: Test that the system works correctly. |

Requirements specifications are the product of the first phase and may be included in the contract as an annex or “statement of requirement” (if available at the time). A system architecture (or Design Document) is the product of the second phase and is unlikely to be available at the time of contracting, or even if it is, it is unlikely to be included except to the extent necessary to identify (for example) hardware items to be purchased.

Conceptually the requirements and design phases can be thought of as separate (although in practice that is not always the case and in fact some iterative software development models assume that one flows into the other)\(^{573}\). The architectural details (that is, how the system accomplishes its tasks) are generally not as important to the customer as the requirements information (that is, what the system is required to do). Of course architectural details are important to the developers because they usually have a direct impact on how the system is implemented. The focus here however is on the customer. The customer is usually mainly concerned with what the system does and how well it performs rather than the internal details of how the software is constructed, except perhaps to know that it has been built to the required quality. Simply put, provided the developer implements a system which meets the specified requirements, then it is of little concern to the customer how the developer does this.

Based on the above, if changes are required to the architecture, then it is likely that the conventional model will be adequate provided that the requirements don’t also change. The reason for this is that changes to the architecture will probably be made for the purpose of implementing the contracted requirements anyway, or for other business

\(^{572}\) Note: that this and following assessments in this document have been made from a general perspective and specific factual situations may naturally result in a different assessment of suitability, when taking all the circumstances into account.

\(^{573}\) See Section: 2.4.2 Lifecycle Models for a discussion on this point.
related reasons such as providing a product which may is more adaptable to other customers' requirements. If the requirements have not changed then there is probably no need to change the contract as architectural changes may not be reflected in it.

Despite this, sometimes architectural details are included into contracts for various reasons. As an example, a contract may specify a particular type of computer hardware for the system to be developed. The reason for this may be simply that the customer already has machines of this type with sufficient spare capacity to run the new application. If this architectural detail changes however then it is likely that contract changes will be required.

Overall, if architectural changes are instigated by the developer for the purpose of achieving existing requirements rather than for the purpose of implementing new or changed requirements, then probably no contract changes should be required. If on the other hand (say) a developer requests architectural changes which in effect constitute new requirements then contractual issues may arise.

Thus a system architecture describes how the system components are connected and what role each component has. Any change to the architecture, for example to improve data throughput or to eliminate bottlenecks may cause the components themselves to change. If the component changes internally, for example to make it more efficient or process data faster, then there is likely to be no problem with the conventional model in that regard. If changes to components require changes to the interface definition between components then in some cases the contractual boundaries between components from different suppliers may be affected.

Thus in large systems development it needs to be remembered that each component of the system will need to conform to particular interface requirements if it is to work with the rest of the system. If the architectural changes affect the interfaces between components then the conventional model may not be entirely adequate because of the issues discussed in the previous Section related to requirements changes.

Once again we see that an ability to cope with change may be crucial to a successful outcome. It is probable that the contract does not include system architectural details to any great extent but rather, defines the requirements of the system in the form of an annexed requirements specification.

Overall, for the reasons given above the:

Conventional model assessed as: Poor to Fair for this criteria.

5.5.3.1.3 What capacity does this relationship model have in dealing with a system which needs to grow?

Once a system has been developed and installed it may need to be upgraded from time to time to cope with such things as an expanding customer base, computer hardware and operating system obsolescence, and compliance with legislative changes (to give some examples).

If a conventional relationship model is used where the customer, as systems integrator contracts for the sub-systems to be integrated then prima facie, under that initial contract
no upgrades or changes are generally possible unless variation clauses are invoked or a separate contract for the work is established.

Depending upon the nature of the changes required, the work may be simply be a matter of upgrading the capacity of the hardware processor speed or storage capacity while keeping the software the same. On the other hand if, for example, the operating system becomes obsolete and is no longer supported by the manufacturer then “porting” the system to a new or upgraded operating system may be a more complex task. During the time between when the system is initially installed, and when the changes are required, it is likely that technology will have advanced and the people familiar with the technical details of how the system works may have also moved on. In addition, if the intellectual property rights of each sub-system are not owned by the customer or licensed to it, then there is effectively a monopoly situation where only the original developers of the sub-system have the legal right and practical skills to make the changes. As a consequence of all these factors any change is likely to be expensive, and possibly time consuming.

Despite this, the conventional model gives the customer, as systems integrator, control over the contracting process to the extent that it can choose when and if it should upgrade. This would probably need to be done however via new contracts with the original developers of each sub-system.

The overall result therefore is that, although changes are possible through a conventional model, they are likely to be expensive and perhaps not very practical in the long term. The systems integrator may be locked into a situation where any upgrades require changes to be made through new contracts with each developer. To mitigate against this situation the systems integrator should ensure that:

- It has sufficient intellectual property rights or licences to reduce the monopoly situation of having only one supplier of each sub-system
- open system interfaces are specified where possible, and
- if necessary and where possible, maintenance and support arrangements are negotiated prior to the signing of the contracts for sub-system acquisition so that a competitive bargaining position is still available.

Overall, for the reasons given above the:

Conventional model assessed as: Poor

for this criteria.

5.5.3.1.4 What capacity does this relationship model have to allow changes to be easily introduced into the system?

To some extent this question is similar to the previous one except that it is focused more on maintenance activities rather than the major upgrading or replacement of components. Maintenance of a system can be (and usually is) the subject of a separate contract apart from the main development or integration contract. To that extent the customer or systems integrator can exercise a degree of control over what maintenance activities are undertaken. In addition, as by this stage the system has already been developed, integrated and installation complete, to some extent is easier to estimate the cost and schedule associated with follow-on maintenance activities.

Despite the above, problems identified in the previous Section also exist however, although primarily in relation to the long term maintenance of a system. Year 2000
problems are a classic case in point. Systems developed using a two digit date field for the year are a problem at the turn of the century if the year "00" is interpreted as anything other than 2000 (for example 1900).

The point is that systems require maintenance for a number of reasons and in particular, "legacy" systems\textsuperscript{574} may be difficult or costly to maintain. One of the reasons for this is that personnel familiar with the system move on and new personnel need to be trained at significant cost and time. In addition, the software development tools and computing platforms may not be available or supported by the manufacturer after a few years elapse.

Overall, with the conventional model, maintenance activities may be undertaken by the establishment of a separate contract. Longer term maintenance may however become costly depending upon the nature of the system because of reasons given above. For systems with larger numbers of installations it may be acceptable for the developer to maintain and upgrade the products at its own cost and then sell the new versions to customers as they become available. This type of arrangement is common with general purpose software such as word-processors, spreadsheets, operating systems etc.

For larger more complex systems with a low number of installations however it is probably more common for "time and materials" type contracts to be put in place for maintenance activities. In some ways a time and materials approach is fair on the parties because it is only the actual cost that is accounted for (plus profit margins) and accordingly any inaccuracy in estimating the total cost of a job has less effect. Despite this in order to minimise adverse effects of maintenance activities the following should be considered:

- Technology refresh. At some period of time the user of the system should consider a technology refresh cycle where obsolete components are upgraded or replaced in their entirety.
- Open systems. Open systems generally aim to make it easier to replace obsolete components with newer technology and hence provide a longer term option, from the maintenance perspective, of giving the customer the choice of replacing obsolete components entirely rather than continuing to maintain an obsolete sub-system using unsupported tools and technologies.
- Continuity of Maintenance. Sufficient continuity of work and staff knowledgeable in the system needs to be maintained in order to support long term maintenance requirements. In a large complex system technical knowledge of how the system works can take a considerable amount of time to develop. In some systems any changes to one part of the system can cause unexpected effects in other parts of the system\textsuperscript{575}.
- The maintenance contracts should be negotiated in conjunction with the main contract for acquisition of each sub-system in order to avoid a monopoly situation of being forced (from a business perspective) to into a maintenance contract with a single service provider.

The conventional model provides more or less a "big bang" approach to systems integration by putting in place specific contracts for the acquisition of major sub-components, longer term maintenance is not necessarily addressed adequately. Unless positive mechanisms are put in place, a simple arrangement based on the conventional contracting model may not adequately address the maintenance issues discussed above.

\textsuperscript{574} which are systems or components developed a significant time ago using technology and techniques which are now out of date by today's standards, but which nonetheless have a crucial part in the business operations

\textsuperscript{575} if the software exhibits a characteristic called "coupling" then changes in one part of the system may affect
Overall, for the reasons given above the:

Conventional model is assessed as: Poor to Fair for this criteria.

5.5.3.2 Micro-agreements in the Context of Software Project Planning

The concept of "Software Project Planning" was discussed in detail in the previous chapter. In summary:

"Software Project Planning involves developing estimates for the work to be performed, establishing the necessary commitments, and defining the plan to perform the work. A plan is established to address the commitments to the customer according to the resources, constraints, and capabilities of the project. The plan provides the basis for initiating the software effort and managing the progress of the work."

5.5.3.2.1 Does the relationship model have the capacity to deal with developments where the schedule is important?

In any engineering contracts, milestones are important to provide a degree of certainty to the parties and to provide a basis for scheduling resources. The problem with software development however is that it can be difficult to estimate the amount of work involved and the timeframe which is required to perform the work. Software projects are notorious for being delivered late and costing more than anticipated and accordingly any schedules or milestones inserted into a contract must have a sound basis for their derivation.

Although the buyer may have rights against the developer if the schedule is not achieved it is presumably preferable, from a business perspective, that the schedule estimates be achievable and based on solid modelling and estimation practices rather than just a guess (which is often what happens). In the previous chapter, Section 4.6: Case Study 2 - Micro-agreements in the Context of Software Project Planning looked at problems associated with poor project planning and listed a number of specific ways in which these problems can be addressed.

In answering this question posed above however a conventional model based on a rigid set of contracts between the customer and individual suppliers has some advantage because it is usual for specific and very clear milestones to be identified in the contract. There is thus a large degree of clarity and certainty in the obligations related to schedule. From that perspective then it appears that the conventional model has a good ability to set clearly defined contractual goals and to identify clear timeframes for the accomplishment of specific tasks. Prima facie then the conventional model should be rated quite highly against this criteria.

Despite the above there are some problems however. As previously mentioned in Chapter 2, estimating the effort and schedule required for a particular project can be difficult. In fact, based on industry experience, it appears to be more likely than not that the effort and schedule estimates in the contract will be wrong, or at least will be underestimated. As other parts as well, sometimes adversely or unexpectedly

578 for example, the initial estimate of days to ship the Microsoft Word for Windows Version 1.0 product was 365 days whereas the actual number of days was 1887: McConnell, Rapid Development - Taming Wild Software Schedules, Microsoft Press, 1996, at p. 208.
an example, McConnell\textsuperscript{579} provides quantitative data that indicates, the \textit{shortest} possible timeframe in which a 40,000 line computer program can be developed is eleven months (for a business product). Having a contract which clearly states that it is to be delivered in less than that time will not in itself cause the system to be developed on time. Moreover it is likely to give the parties a false sense of security at the start of the contract and send them into panic mode towards the end.

The point is that if accurate estimates of effort and schedule are available then the conventional contracting model is a good approach because it can provide a large amount of certainty to the parties. It can also provide feedback on progress by specifying intermediate milestones or deliveries.

If however the effort and schedule estimates are not accurate then specifying them in a contract using a conventional relationship model will not necessarily help. If the schedule really is important to the customer then more is needed. Specifically, if the conventional model is used then there needs to be a solid basis for the schedule estimates used, coupled with regular feedback on progress and possibly a mechanism to adjust the relative priority of functionality so that decisions can be made in relation to product capability and schedule timeframes.

Overall, for the reasons given above the:

Conventional model is assessed as: Poor to Fair
for this criteria.

5.5.3.3 Micro-agreements in the Context of Software Project Tracking and Oversight

The concept of "Software Project Tracking and Oversight" was discussed in detail in the previous chapter. In summary:

"Software Project Tracking and Oversight involves tracking and reviewing the software accomplishments and results against documented estimates, commitments, and plans, and adjusting these based on actual accomplishments and results. A documented plan for the software effort is used as the basis for tracking the software activities, communicating status, and revising plans. The software activities are monitored by the software managers on a regular basis. Regular technical reviews and reviews with the project manager and senior management are conducted to ensure that management and staff are aware of the software status and plans, and that issues receive appropriate attention."\textsuperscript{580}

5.5.3.3.1 Are the overheads low for this relationship model?

With the Conventional model the "overheads" can be low when compared to some of the more sophisticated relationship models discussed later. In order to illustrate the reasons for this, as an example, the Captive model (as discussed in detail in a following Section of this chapter) has at its basis, the formation of a separate company. Overheads associated with the administration of that company are likely to be higher than for the administration of a simple contract (such as is used in the conventional model).

Overall, for the reasons given above the:

Conventional model is assessed as: Very Good

for this criteria.

5.5.3.3.2 Does this relationship model give the customer good visibility of progress?

In theory the conventional relationship model can give the customer good visibility of progress but in practice this is not always the case. A conventional model will usually ensure that contractual milestones are clearly spelt out and to that extent, if a milestone is not achieved, the customer gets feedback on progress. For example, if one of the milestones identified in the contract is to successfully complete the integration testing of one component by a particular date, then naturally if that activity is not complete by that date then the customer gets a clear indication of a possible schedule slippage.

To that end the more detailed the list of contractual milestones the more feedback the customer can formally get of progress that has been made on the project. Contractual milestones can thus be a powerful tool which focus’s the parties efforts in the achievement of particular tasks. If a milestone is missed then it may signal to the customer that some underlying problems exist and longer term schedules are at some risk.

Despite the above the problem with contractual milestones is that the level of visibility to the customer is restricted (in theory) to a relatively low number of specific (but important) milestones when compared to the totality of activities which occur during the life of a large project. The visibility which the customer gets may be thus limited to a sample of (say) 10-20 specific points in the life of a project consisting of many hundred or thousands of separately identifiable activities. Because of this low sampling rate it is critical that any milestones which are identified in the contract be effective in identifying to the customer the extent of schedule and cost variances against planned targets. Depending on the type of milestones it may be difficult if not impossible to accurately assess whether the schedule is on track by using these milestones alone. For example, the delivery of a “Software Design Document” may itself constitute a milestone which is met by the contractor, but without some assessment of the quality of the product delivered, it may even lead the customer into a false sense of security if, despite being “delivered”, it is inadequate for the purpose intended.

The planning and identification of milestones is of fundamental importance to any software project but to rely on contractual milestones as the sole means of tracking progress may not be enough. Large government projects often take the milestone approach further by requiring the contractor to submit Cost Schedule Control System581 (CSCS) reports on a regular basis. These are intended to provide the customer with a detailed breakdown on activities which aims to provide information on any cost and schedule variations to the plan. This approach can be useful but depending upon the size of the project can be complex to implement and require sophisticated tools to be effective.

Overall, it is considered that the conventional relationship model gives some ability to provide the customer with progress visibility but is more likely than not to be restricted to a limited number of progress milestones points. At best the achievement of milestones gives some indication of satisfactory progress but there is a danger that the milestones, if relied upon inappropriately, can lead the parties into a false sense of security.

Overall, for the reasons given above the: Conventional model is assessed as: Poor to Fair

for this criteria.

---

581 Various standards for Cost Schedule Control Systems are in existence. The United States Department of Defence: Cost Schedule Control Systems, Instruction 5000.2 is one example.
5.5.3.3 Does the relationship model give management good visibility of progress?

In order to answer this question we need to establish who "management" is in the context of this particular relationship model. As previously discussed, with the conventional relationship model, the customer itself acts as the systems integrator and as such takes on a managerial role in ensuring that system wide goals are achieved. The development and supply of individual components of the system however are more likely to be managed by the suppliers directly.

So assuming that the customer, in this relationship model, also has the task of managing the project to ensure that the system is integrated and delivered on-time, on-budget and to the required quality, what visibility of progress does it have?

In this model, since the customer takes on the system management role, the answer is the same as for the previous question. That is, in theory there is good visibility through the use of milestones but in practice this may come down to a relatively small set of milestones. These are likely to give some indication of progress of course but depending on the granularity of these milestones any indication of schedule slippages or cost over-runs may come too late.

Overall, for the reasons given above the:

Conventional model is assessed as: Poor to Fair for this criteria.

5.5.3.4 Micro-agreements in the Context of Sub-Contractor Management

The concept of "Sub-Contractor Management" was discussed in detail in the previous chapter. In summary:

"Software Subcontractor Management involves selecting a software subcontractor, establishing commitments with the subcontractor on the work to be performed, coordinating activities with the subcontractor, and tracking and reviewing the subcontractor's performance and results. The subcontractor is selected on its ability to perform the work. A documented agreement covering the technical and non-technical (e.g. legal, financial and administrative) requirements is established and is the basis for managing the subcontract. Regular technical and management reviews are conducted to ensure that management and staff of both organisations are aware of the software status and plans, and that issues receive appropriate attention." 562

5.5.3.4.1 Does the relationship model allow the management of risks effectively?

In order to answer this question it is important to discuss briefly the types of risk associated with large developments. As a starting point it is likely that, in the broadest sense the important factors are schedule, cost, and quality.

To some extent these three factors can be balanced and traded off against each other. For example, if more funding is spent on testing then the quality may be improved but overall the schedule may be delayed to achieve this.

Risks associated with any of the above areas, that is, cost, schedule and quality can lead to problems in the other areas as well. If the project is delayed for any reason then costs will usually be higher because of the need to retain staff and equipment beyond the expected timeframes.

Although the above discussion focuses on three primary risk areas: cost, schedule and quality, commentators have identified other groupings. For example, McConnell\textsuperscript{583} provides a reasonably comprehensive list of over 100 possible risks associated with software development projects, and these risks are grouped under the following headings: schedule creation, organisation and management, development environment, end-users, customer, contractors, requirements, product, external environment, personnel, design and implementation, process. Some examples of the specific risks identified within these groups include:

*Schedule is optimistic rather than realistic*; *unfamiliar areas of the product take more time to design and implement*; *inefficient team structure reduces productivity; planning is too poor to support the desired development speed*; *facilities are not available on time*; *end-users insist on new requirements*; *customer micro-manages the development process resulting in slower progress than planned*; *contractor does not deliver components when promised*; *requirements are poorly defined*; *unacceptably low product quality*; *low motivation and morale reduce productivity*.

Many more examples of possible risks can be found in the literature\textsuperscript{584} however the issue is firstly how these risks can be mitigated, and secondly, whether this relationship model provides a structure suitable for risk mitigation generally.

Essentially the key to managing risk is flexibility. In a complex development the nature of the work can be difficult to predict and accordingly the risks associated with the development may also need to be addressed by some flexibility in the way that the projects are run.

In a contracting sense there are basically two types of risks (from the point of view of the present discussion) – those that cross contractual or organisational boundaries and those that do not. For example, project managers should be able to mitigate the risk of “porting” (i.e. moving) new software to a new operating system without having to raise contractual


\textsuperscript{584} ibid
issues. On the other hand if the contract specifies a particular interface between software components, then the mitigation of that risk may affect more than one party – i.e. the developers of the respective components.

In respect of the conventional relationship model it is submitted that any risks which cross contractual or organisational boundaries may not be easy to resolve or mitigate against without formal contract changes. As an example, if the customer has a contract with a developer for the supply of a component of the system, then if the interface specification between components of that system changes, the contract will also need to be modified to take into account those changes. Whether this is a good or bad thing depends upon the circumstances of each case.

As discussed in an earlier chapter one of the characteristics associated with a large development is the amount of change involved. In order to mitigate risk the relationship model adopted needs to be able to adapt to such change. The conventional relationship model in some ways can be somewhat rigid in that the rights and obligations of the parties are (in theory) fully specified up front in the contract between the systems integrator and the supplier of particular components. Particularly in large system acquisitions it is important to understand that attempting to define the full scope of work early in the development cycle may not be feasible and any attempt to do so may provide a false sense of security to the parties and unrealistic expectations.

Overall, for the reasons given above the:

Conventional model is assessed as: Poor for this criteria.

5.5.3.5 Micro-agreements in the Context of Software Quality Assurance

The concept of "Software Quality Assurance" was discussed in detail in the previous chapter. In summary:

"Software Quality Assurance involves reviewing and auditing the software products and activities to ensure that they comply with the applicable processes, standards, and procedures, and providing the staff and managers with the results of their reviews and audits. The software quality assurance function is required on all projects. The group performing this function is independent of the software groups and project management. A senior manager who is committed to handling all major software quality assurance issues is identified. Where compliance issues exist, the software quality assurance group works with the appropriate managers, including senior management where required, to resolve the issues."

5.5.3.5.1 Does the relationship model have the capacity to deal with developments where system reliability is important?

Why is this question relevant? Many attributes are available to define quality and reliability is just one of these. The question posed relates to how well the relationship model is in dealing with quality issues. The "reliability" attribute has been selected for the purposes of discussion as one example of an important quality attribute. Other quality factors such as portability, interoperability, maintainability, etc. could also have been used as an example.

---

So in answering the question posed, it is considered that the conventional relationship model does have a good capacity to deal with developments where system reliability is important provided that there is some measurable criteria for evaluation. Measures of reliability are sometimes easier to establish and write into a specification than some other more abstract quality measures. For example, suitable reliability could be defined as a Mean Time Between Failure (MTBF) of 1,000 hours. This specification could be measured by (for example) running the system continuously for a period of time and recording failures. These failures could then be analysed statistically to produce the desired measure.

So how does the conventional mode deal with developments where system reliability is important? If there is a measurable quality criteria that can be specified in the contract then it is submitted that the conventional model may be adequate. This is because the contract itself would clearly provide the quality required in the product, in a way that was measurable and quantifiable. In the case of "reliability" for example, the conventional model may be adequate because the mean time between failure requirements can be specified in the contract between the systems integrator and the developer of particular components.

In terms of a software engineering process, it is possible to specify that each contractor conform to a particular quality standard such as IEEE Std 1298-1992; AS 3563.1-1991 for example. The advantage of this approach is that presumably a third party accreditation of a supplier’s quality system has been undertaken. In practice however it may be necessary to ensure that the standard is not called up 'blindly' but rather, that the relevant parts of the standard are specified adequately.

In summary, in answering the question posed above, the conventional relationship model is considered to have a good capacity to deal with quality factors under certain conditions. Provided that (ideally) a quality process is enforced upon sub-contractors, and also that measurable quality goals are specified in the contract then some benefit will hopefully arise (although possibly at additional cost). Despite this, once the sub-contracts are in place then there is little capacity to modify the required quality factors without changing the contract.

Overall, for the reasons given above the:

Conventional model is assessed as: Fair to Good for this criteria.

5.5.3.5.2 Does this relationship model require sophisticated developers and management?

The reason for asking this question is that complex technical undertakings may require sophisticated development and management practices. In a conventional relationship model the customer itself takes on the role of systems integrator and accordingly needs an adequate capability to understand and control the system level architecture. The customer, using this model, needs to fully understand the system requirements (including all functionality, performances and interfaces), and needs to be able to adequately specify these in contractual terms appropriate to sub-contractors. After sub-contracts are in place then the ability to cope with change is to some extent limited by the ability to implement contract changes. Commercially any contract changes may be costly simply because the customer may be locked into a particular supplier for those changes and hence will effectively be in a monopolistic situation.
Overall, the conventional mode is likely to require sophisticated developers and managers and particularly the customer itself when the systems being developed are complex or large.

Overall, for the reasons given above the:

Conventional model is assessed as: Poor to Fair – very sophisticated for this criteria.

5.5.3.6 Micro-agreements in the Context of Software Configuration Management

The concept of “Software Configuration Management” was discussed in detail in the previous chapter. In summary:

"Software Configuration Management involves selecting project baseline items (e.g. the project description, products, and process specifications of the project), controlling these items and changes to them, and recording and reporting status and change activity for these items. Changes to these baseline items are controlled systematically using a defined change control process. The configuration (software or documentation) of a system, or of any of the controlled intermediate or support products, can be distinctly identified at any point in time." 586

5.5.3.6.1 Does the relationship model have the capacity to enforce good software configuration management practices?

The configuration management process is an important function but conceptually reasonably straightforward. It is essentially a set of processes designed to allow changes to configured items (such as source code) in a controlled manner. Provided that either the developer of a particular component follows a defined configuration management process and uses appropriate tools then for that component there should not be major issues. To reduce the risk and increase confidence it could be made a condition of the contract that a configuration management process be in place. Standards are available which can be used as a guide and some of these are listed in Appendix A. Software Standards.

Despite the above, system wide configuration management practices are directly under the control of the customer (as systems integrator) when the conventional relationship model is used. The customer itself needs to ensure it has adequate configuration management processes in place to manage the acquisition of sub-components and their integration into the system as a whole.

Thus in summary, the conventional model is adequate for enforcing good configuration management practices provided that adequate processes are in place and preferably required as a contractual obligation, and in addition the customer, as systems integrator, enforces good configuration management practices on a system wide level.

Overall, for the reasons given above the:

Conventional model is assessed as: Good for this criteria.

5.5.3.7 Assessment of Legal Risks

What capacity does this relationship model have in minimising legal risks? Specifically, what is the likelihood of claims under contract, estoppel, trade practices or other legal doctrines if this relationship model is used?

In the conventional model, the customer itself takes on the system integration role and enters into contracts with suppliers for delivery of components of the system. Accordingly, the basis of the relationships in this model generally centre around contracts for the delivery of components of the system, and associated services. In order to assess whether there are any significant legal risks associated with this model, a starting point should therefore be the likelihood of breaching these contractual obligations.

Traditionally, contracts used in the conventional model often include (or reference) a technical specification which sets out the requirements of the items to be delivered. These specifications generally attempt to define in detail the functionality, performance, and interfaces required of the software. In addition, other contractual clauses also identify price, schedule and quality issues. Overall the purpose is to provide a degree of certainty to the parties and to identify their respective rights and obligations under the contract.

Despite this it is likely that requirements will change during the life of the project and invariably these contracts provide a number of ways of coping with change including such mechanisms as variation clauses and dispute resolution processes. These types of mechanisms however tend towards fixing the problems after they occur rather than preventing them from occurring in the first place. Other ways of introducing flexibility into the development exist including "time and materials" type contracts where sub-contractors effectively work under the direction and control of a prime contractor rather than having the sole responsibility of delivering components itself. This type of contract however, while obviously acceptable in some cases, may increase the uncertainty in the final cost of the system.

Problems occur because, on the one hand, the parties need contractual certainty, while on the other history has shown that software development activities, and particularly those associated with the development of large systems, are likely to be dynamic in nature and require changes throughout the life of the system.

Accordingly one major risk area associated with the conventional model is the potential for actions in breach of contract.

In respect of some of the other possible actions, an example of a development undertaken using the conventional model is presented in the previous chapter in Case Study 1 (See Section 0 above). In that case study a number of legal issues were examined including: misrepresentation (common law), misrepresentation (Trade Practices Act, 1974), breach of contract (including breach of implied terms), estoppel, economic duress, unconscionable transactions (Trade Practices Act, 1974). In general it was found under the particular fact situation that claims could possibly be made in a number of these areas. In summary however, although the particular fact situation will naturally dictate whether a claim may be available, in general it is submitted that the conventional relationship model may increase legal risk in a number of areas (compared to other models) including:

- the risk of actions for breach of contract may be increased (as discussed above);
- the conventional model encourages the formation of contracts for information technology products when industry experience shows that it may be difficult to

---

567 Section 2.5 Requirements Management and Section 0 Case Study 1 - Micro-agreements in the Context of Requirements Management discuss this.
estimate the cost and schedule associated with acquiring these products. Any estimates of cost, schedule or performance of the system which therefore turn out to be incorrect, may give rise to actions in misrepresentation, and estoppel for example\(^{586}\).

Overall, for the reasons given above:

The Conventional Model is assessed as poor in minimising legal risks particularly in relation to breach of contract actions, and to a lesser extent, in relation to misrepresentation and estoppel.

### 5.5.3.8 Summary of Assessment – Conventional Model

The following table summarises the above assessment.

<table>
<thead>
<tr>
<th>Relationship Model Evaluation Criteria</th>
<th>Effectiveness of Relationship Model in coping with micro-agreements</th>
</tr>
</thead>
</table>
| Micro-agreements in the context of Requirements Management | • Does the relationship model have a capacity to deal with inadequately defined requirements?  
Poor  
(see 5.5.3.1.1) |
| | • Does the relationship model have a capacity to deal with an evolving system architecture?  
Poor to Fair  
(see 5.5.3.1.2) |
| | • Does the relationship model have a capacity to deal with a system which needs to grow?  
Poor  
(see 0) |
| | • Does this relationship model have the capacity to allow changes to be easily introduced into the system?  
Poor to fair  
(See 0) |
| Micro-agreements in the context of Software Project Planning | • Does the relationship model have the capacity to deal with developments where the schedule is important?  
Poor to fair  
(See 5.5.3.2.1) |
| Micro-agreements in the context of Software Tracking and Oversight | • Are the overheads low for this relationship model?  
Very Good  
(See 5.5.3.3.1) |
| | • Does this relationship model give the customer good visibility of progress?  
Poor to Fair  
(See 5.5.3.3.2) |
| | • Does the relationship model give management good visibility of progress?  
Poor to Fair  
(See 5.5.3.3.3) |
| Micro-agreements in the context of Sub-contractor Management | • Does the relationship model allow the management of risks effectively?  
Poor  
(See 5.5.3.4.1) |

\(^{586}\) In Case Study 1 (See Section 0 above)

---

Paul P. Parnell 219
<table>
<thead>
<tr>
<th>Micro-agreements in the context of Software Quality Assurance</th>
<th>• Does the relationship model have the capacity to deal with developments where system reliability is important?</th>
<th>Fair to good (See 5.5.3.5.1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Does this relationship model require sophisticated developers and management?</td>
<td>Poor to Fair – yes, very sophisticated (See 5.5.3.5.2)</td>
<td></td>
</tr>
<tr>
<td>Micro-agreements in the context of Software Configuration Management</td>
<td>• Does the relationship model have the capacity to enforce good software configuration management practices?</td>
<td>Good (See 5.6.3.6.1)</td>
</tr>
<tr>
<td>Legal Risks</td>
<td>• What capacity does this relationship model have in minimising legal risks?</td>
<td>Poor (See 5.6.3.7)</td>
</tr>
</tbody>
</table>

**Figure 32 - Evaluation of Conventional Model Attributes**

**5.5.3.9 Looking at trends – Micro-agreements and the Conventional Relationship Model**

Overall, when a micro-agreement occurs within the context of the software engineering fundamental "best practice" areas discussed above, then it is assessed that:

**Key Findings**

**Finding 1:** The conventional relationship model is likely to allow micro-agreements to occur, or at least not prevent them from occurring.

**Finding 2:** The conventional relationship model is likely to provide little flexibility to enable changing circumstances to be managed effectively, except by either direct contract change amendments, or by appropriate software engineering processes which are contractually supported.

**Finding 3:** The conventional relationship model is likely to allow the parties to a contract little discretion to accept and implement "good" micro-agreements (that is, those which are consistent with the overall business objectives of the parties), except by either direct contract change amendments, or by appropriate software engineering processes which are contractually supported.

**Finding 4:** The conventional relationship model is likely to allow the parties to a contract the discretion to reject and not implement "bad" micro-agreements (that is, those which are not consistent with the overall business objectives of the parties). The main reason for this is that prima facie they are not contractually binding. Despite this, issues such as estoppel, unconscionability, etc. may be relevant.
Finding 5: Particularly for large systems, the conventional relationship model is likely to result in an information technology system which is less likely to satisfy customer needs in terms of cost, quality and schedule, unless a solid software engineering process is in place, preferably as a contractual requirement on the parties.

Finding 6: The legal risk of breach of contract is relatively high for the conventional model (compared with some other models) especially if changing requirements cannot be introduced easily into the contracting process.

Finding 7: The conventional model encourages the formation of contracts for information technology products when industry experience shows that it may be difficult to estimate the cost and schedule associated with acquiring these products. Any estimates of cost, schedule or performance of the system which therefore turn out to be incorrect, may possibly lead to actions in misrepresentation, and estoppel (for example).

Finding 8: The conventional relationship model may be the best choice for simpler information technology developments, or ones where there are few parties involved, or where the requirements are well known and fully specified. Regardless, it is considered that the best chance of achieving success with this model will occur if appropriate software engineering processes are required as contractual obligations.

5.5.3.10 Conclusion
The convention model is one where the customer effectively acts as the systems integrator itself and obtains services from third parties as required. There are a number of advantages and disadvantages to this approach however in summary, based on information distilled from the above assessment of the model, in order to successfully implement this model:

- the customer needs to fully understand the system requirements
- the customer needs a clearly defined system architecture to be able to fit new components into a system
- If the requirements are fully understood and the specification is complete, then if software engineering process formalities are adhered to a high quality and reliable system should be achievable
- risks can be difficult to accommodate within the one contract (and may require additional contracts or “cost-plus” types of arrangement) to cover changing requirements
- the model doesn’t easily allow for corrections half way through unless additional contracts are put in place

Paul P. Parnell
• if schedule is important then this model can provide for specific milestones in the contracts

Of fundamental importance to the successful use of this model is the necessity for the customer to fully understand and document the requirements. Any uncertainty in specifications provided to sub-contractors will increase the risk of components not interacting together successfully. Specifically, a detailed specification should be provided to each of the subcontractors identifying technical aspects such as the functionality, performance and interfaces of the system, however it also needs to be recognised that in a large system these requirements are likely to change. A mechanism for coping with this change is essential.

One advantage of this model is however that the customer, as systems integrator, has full control over the contracting process as it is free to contract with whomever and under whatever terms are commercially acceptable to the parties.

Overall, this model has the advantage of being relatively simple to understand, as well as providing some contractual certainty over cost, schedule and quality and giving the buyer some control over sub-contracts. On the down-side however, it requires that the customer has an in-depth knowledge of system requirements and technical architectural issues, and also that any errors or omissions in contracted specifications may be difficult to rectify without the overheads of specific contractual amendments.
5.6 SERVICE MODEL

5.6.1 Description
In this model the customer identifies and uses a specialist systems integrator to control and oversee all development activities. The systems integrator sub-contracts products and services as necessary for the development and integration of the system.\footnote{Based on a brief description provided in: Pollard, S. "Managing Risk in Major IT Projects", 1996, http://www.gtlaw.com.au/gt/bin/frameup.cgi/gt/pubs/mannrisk.html}

5.6.2 Example
Based on the example used to illustrate the previous model, if a customer required the development of a flight simulator for training crews, then it could possibly use the service model to acquire the system. A systems integrator would be identified and contracted to deliver the fully integrated system and any sub-components of the system would be acquired by the systems integrator itself and fully integrated prior to delivery to the customer. The system may require, for example integration of two different navigation sub-systems (for redundancy) such as a satellite based Global Positioning System (GPS) sensor together with an Inertial Navigation System (INS). As these two sources of navigation data need to be integrated into the flight simulation model the types of data, data rates etc. need to be consistent. In this situation, the systems integrator could contract with the suppliers of each of the two navigation systems directly to provide the required hardware and software, and once fully integrated, the system could be delivered to the customer.

5.6.3 Analysing this Relationship Model
Based on the Method of analysis provided above in Section 5.4.2, the starting point for analysis of this model is to address the evaluation criteria provided in that Section.

5.6.3.1 Micro-agreements in the context of Requirements Management

5.6.3.1.1 What capacity does this relationship model have in dealing with inadequately defined requirements?
The answer to this question is that it depends on who’s point of view we are looking from. From the customers point of view probably the reason for using a systems integrator in the first place is that the customer itself does not have the resources, or technical or managerial expertise to undertake the task and accordingly contracted a specialist systems integrator to perform this work. It is likely that any contract with a systems integrator would thus be written in broad terms and that the systems integrator itself may be required to develop the detailed technical specifications in consultation with the customer. If that is the case then any requirements changes which do not affect the broad
scope of the systems integrator’s task are likely to fall on the systems integrator to resolve and accordingly the customer may not necessarily be affected.

As an example, in the fact situation given above, if the customer simply required a “fully integrated and redundant navigation system” then it may be implied that the accuracy of aircraft position and regularity of position updates be in accordance with recognised international flight navigation standards. Any deviations from those standards by subcontractors supplying would require management by the systems integrator to rectify.

Thus, from the point of view of the customer it is probably in its expectation (and possibly an implied or explicit term of ant contract it has with the systems integrator) that the detailed engineering characteristics of the system would be managed by the systems integrator. Any change therefore to requirements related to such issues should be relatively “invisible” to the customer.

Despite this it is also a fact of commercial life that sometimes a systems integrator can use a vague or broadly defined customer specification to its own advantage particularly where tight budgets and timeframes are involved. Naturally, if requirements change to the extent that the customer requires significantly new or unplanned system capabilities then the systems integrator should not be entirely responsible for implementation without appropriate reward. It is important in such contracts therefore to ensure that firstly, the scope of the work is as well defined as possible; and secondly, that the systems integrator is contractually required to use professional skills and judgement and to identify and implement detailed system requirements beyond those specifically listed in the contract between the customer and the systems integrator.

Overall, for the reasons given above the:

Service model is assessed as: Fair
for this criteria.

5.6.3.1.2 What capacity does this relationship model have in dealing with an evolving system architecture?

From the customers point of view, generally speaking the architectural details of the system do not matter, provided that the system is fit for the purpose intended. The customer is interested in obtaining delivery of a system that performs particular specified functions. How the system performs those functions is usually of lesser importance to the customer and in fact, unless there are some overriding requirements such as using existing customer hardware, it is preferable that the systems integrator is free to make architectural decisions based on solid engineering analysis rather than being forced into using (possibly) inappropriate components specified by the customer in the initial contract.

Thus in answering this question, any issues related to system architecture (that is, simply put, how the system performs its tasks) are probably of little concern to the customer. Of course the systems integrator has a prime role of ensuring all systems components operate effectively together and thus any architectural changes are of paramount importance to it.

Overall, for the reasons given above the:

Service model is assessed as: Good
for this criteria.
5.6.3.1.3 What capacity does this relationship model have in dealing with a system which needs to grow?

Once the contract between the systems integrator and the customer is in place then what flexibility for change is there using this model? Naturally this will depend upon the particular circumstances but in general, the earlier any changes are identified the easier and presumably less costly they will be to implement. In a larger project the systems integrator will be keen to define the full scope of work early and let sub-contracts for the development of components of the system. After a period of time therefore it is likely that a set of contracts will be in place which cover development and delivery of individual components as well as for the integration of the system as a whole.

What happens then if the system needs to grow? As an example, what if half way through the development cycle a competitor develops a competing product with extra features? The simple answer is that new contracts could be written to cover changes to the system being developed. In practice however this may be difficult or costly to achieve when the customer is now effectively “locked” into a single systems integrator who has also “locked in” sub-contractors to deliver products and services by (presumably) specified dates. If the original process of forming each contract was competitive in the sense that (for example) the customer could have gone to other systems integrators to get the work done, then it is likely that any additional contract or changes made to contracts with existing suppliers would be made under monopoly conditions and possibly with inflated prices.

The key issue is that in this model, the benefit to the systems integrator clearly arises after the initial contracts are in place in that the systems integrator has effectively a monopoly position with the customer. In addition, there may be additional risk involved in the sense that there is little incentive for the systems integrator to co-operate with the acquirer unless financially very attractive.

Overall, one the initial set of contracts are in place, then using this model, it may is difficult to evolve the system or make changes except under this monopolistic situation.

Overall, for the reasons given above the:

Service model is assessed as: Poor
for this criteria.

5.6.3.1.4 What capacity does this relationship model have to allow changes to be easily introduced into the system?

This focus of this question is on whether changes can be “easily introduced” into the system. From a contractual standpoint it is likely that once a set of contracts has been established from the customer to the systems integrator to the sub-contractors then any changes may be very difficult or costly to implement. The reasons for this have been discussed in previous questions however in summary, for a large system in particular, any change is likely to occur under a monopolistic situation. If however the change can be isolated to a single component of the system then it may be the case that only the developer of that component needs to make changes.

If on the other hand the change is one which affects the software “interface” between two or more components, or functionality covered by two or more components then the task of making the change becomes more difficult.

Thus changes made late in the development and integration cycle are likely to be more costly because of the monopolistic situation which is effectively in place. As we have seen
earlier, quantitative data available in the industry indicates that the cost of fixing problems late in the project can be 200 times greater than if the same changes were identified and fixed earlier on\[590\]. Thus changes made very early in the life of the project can, from a technical point of view, cost less than those identified later in the development cycle. When coupled with the Service Model type of relationship however it is likely that even early in the project the cost of making changes may be relatively high because the acquirer is “locked in” to particular suppliers by that stage.

It appears overall therefore that this model does not provide much help in being able to easily make changes to the software although naturally any change is possible depending upon the negotiated agreement of the parties. This model is likely to allow changes to be made but at possibly higher costs than some of the other models. The payment of additional costs by the acquirer may be acceptable under some circumstances, particularly if the systems integrator takes on the risks associated with making the changes. If it is important for the customer to minimise its risk by contracting specialist services then a service model type of relationship may be appropriate.

Overall, for the reasons given above the:

Service model is assessed as: Poor to fair for this criteria.

5.6.3.2 Micro-agreements in the Context of Software Project Planning

5.6.3.2.1 Does the relationship model have the capacity to deal with developments where the schedule is important?

With this model, the acquirer contracts with the systems integrator to deliver the system presumably by a particular date. Similarly the systems integrator sub-contracts with suppliers of products and services related to particular aspects of the system. From the purely contractual picture therefore it appears that there is are large degree of certainty established in the delivery dates. Each of the contracting parties would appear to have entered into its contract with its “eyes open” and presumably would have conducted sufficient analysis and investigation of the tasks required to be able to meet the contracted delivery dates. These contracts should thus provide all parties with a reasonable degree of certainty of schedule.

Why then are so many large software projects delivered late? “Most projects overshoot their estimated schedules by anywhere from 25 to 100 percent”\[591\]. Traditionally it has always been difficult to accurately estimate schedules associated with large software projects and the fact that a legally binding contract is in place does not really change this situation. What it does of course is provide an incentive for the parties to meet the agreed schedules but the schedules may simply be wrong in the first place. At worst, it can provide a false sense of security to the customer that the system will be delivered on time.

Thus the Service model in some ways provides certainty of schedule through clearly contracted deadlines but in other ways may give a false sense of security to the parties that an unrealistic schedule is achievable. If a contracted schedule turns out to be not achievable then the ideal situation would be to have an early warning of this so that corrective action can be taken as appropriate. Early warning of schedule issues may possibly be achieved through regular reviews of progress and an effective software engineering process associated with tracking progress (as discussed in Section 4.7: Case Study 3 - Micro-agreements in the Context of Software Project Tracking and Oversight).

\[591\] McConnell, S. Rapid Development – Taming Wild Software Schedules, Microsoft Press, 199, page 163
Overall, for the reasons given above the:

| Service model is assessed as: | Poor to Fair for this criteria. |

**5.6.3.3 Micro-agreements in the Context of Software Project Tracking and Oversight**

**5.6.3.3.1 Are the overheads low for this relationship model?**

For the acquirer of the system it is probable that the management overheads will be low because it has contracted a systems integrator to perform the work of building the system. The acquirer theoretically would rely to a significant degree on the systems integrator to manage all of the day-to-day tasks associated with specifying, designing, implementing, integrating and verifying acceptability of the system. The systems integrator is likely to sub-contract major portions of the work to achieve these goals and manage to overall system development.

Thus from this perspective the Service Model has relatively low managerial or technical overheads although naturally costs may be greater because of the risks taken on by the systems integrator.

Overall, for the reasons given above the:

| Service model is assessed as: | Fair to Good for this criteria. |

**5.6.3.3.2 Does this relationship model give the customer good visibility of progress?**

The service model approach is likely to only give visibility of progress to the extent which has been contractually provided by way of scheduled progress reviews etc. Thus if the contract between the acquirer and the systems integrator required that monthly progress reviews be held, then from that perspective the acquirer may only get to see the "glossy" presentations made formally each month but in reality not get sufficient visibility of actual progress to ascertain whether the project is on track.

Apart from the usual progress meetings, formal tracking and oversight activities in the contract may be useful to the acquirer in obtaining visibility of progress. In reality however as large system development and integration projects are likely to involve a number of contracts and sub-contracts when this relationship model is used, the extent of interaction between the parties may make it difficult to understand the totality of the work undertaken and thus whether or not it is on track.

Overall, for the reasons given above the:

| Service model is assessed as: | Poor for this criteria. |

**5.6.3.3.3 Does the relationship model give management good visibility of progress?**

Management in the case of the service model is undertaken at a number of levels. The acquirer of the system manages the contractual arrangements with the systems integrator. The systems integrator manages the relationship with individual sub-contractors to ensure components of the system are developed according to requirements
of schedule, cost and quality. The systems integrator also manages the technical aspects of integrating all the sub-components of the system together to ensure they all work together correctly as an integrated unit. Finally, each sub-contractor manages the development of particular system components for supply to the systems integrator.

From the customers point of view the service model provides a level of management visibility commensurate with the contractual clauses in place in the contract between the customer and the systems integrator. Thus if that contract requires monthly progress reviews to be held then the acquirer gets visibility at that level. Other opportunities for management visibility which are contractually based may also include access to draft documentation, achievement of specific contractual milestones, attendance at component level and system test activities, and the placement of engineering representatives on-site at the premises where the technical work is being conducted. Apart from this there is likely to be little visibility otherwise. In fact it may be the opposite case that, for example, if progress payments are linked to specific milestones, a positive (but unrealistic) view of progress may be presented so that progress payments can be claimed.

For the systems integrator, the situation is similar to that of the customer although as it is "closer to the action" via its sub-contracts, it may have more data available to assess risks. For example, the systems integrator may attend monthly progress meetings for each of a number of subcontractors. Even so, it is likely that the amount of data may be limited by that defined in the contract. Certainly any particular sub-contractor will be likely to want to restrict access to data as much as possible to protect its intellectual property, and to that end there is little incentive provided outside contract to provide additional visibility, apart from some "business" issues which include giving access for the purpose of enhancing reputation and building customer confidence.

In answering this question therefore it is likely that management will have visibility of progress, risks and other issues commensurate with the contractually defined access to relevant information. Apart from that it is unlikely that the service model, in itself, will provide or allow any better visibility.

Overall, for the reasons given above the:

Service model is assessed as: Fair
for this criteria.

5.6.3.4 Micro-agreements in the Context of Sub-Contractor Management

5.6.3.4.1 Does the relationship model allow the management of risks effectively?
From the acquirer's point of view probably one of the main reasons for using the service model and appointing a systems integrator is to manage the risks associated with the project. Systems integration work can be complex for large or technically innovative developments and hence risks can also be significant. The risk management in this model can focus on appointing a specialist systems integrator to provide all services associated with the delivery of a system on-time, on budget and to the required quality. Accordingly, the service model does provide a good ability to manage risks from that perspective.

From the systems integrator point of view however all major risk is placed upon it to a large extent. The systems integration contract on the one hand is likely to require the systems integrator to deliver a fully integrated system while at the same time it wears the responsibility of choosing and managing appropriate sub-contractors to supply specialised products and services. Despite that however it is also likely that the systems integrator is
fully aware of the risks involved and that it is responsible for delivery of the integrated system to the customer.

Overall, for the reasons given above the:

Service model is assessed as: Good for this criteria.

5.6.3.5 Micro-agreements in the Context of Software Quality Assurance

5.6.3.5.1 Does the relationship model have the capacity to deal with developments where system reliability is important?

Reliability can be thought of as one particular quality attribute which is likely to be important to any large system. Other quality attributes such as scalability, portability, and user interface consistency are also likely to be important also however the reliability attribute is used as one example for the purposes of illustration of this criteria.

From the acquirer's perspective, when using this model two things are possible to maximise the chance of achieving a reliable system. Firstly, reliability may be specified up front in any specifications which are developed prior to contracting with the systems integrator. Reliability may for example be specified using such measures as "Mean Time Between Failure" and "Mean Time to Repair". The problem with this approach is however that using the service model, defining the requirements (including reliability requirements) will be one of the main tasks of the systems integrator, possibly after the contract is in place between the systems integrator and the acquirer. It may be difficult or unreasonable therefore to have such detail in the initial contract and accordingly, any such figures which are later derived may not be contractually enforceable.

One other option is to specify that the systems integrator be accredited to an appropriate quality standard. This should ensure that at least to some extent a process is in place during the development of the systems which enforces some quality control onto the products in question.

In summary however the service model, in itself, is unlikely to ensure that a reliable product is delivered, except to the extent identified in the contract either by direct specification of reliability requirements, or indirect specification of quality accreditation. In addition the service model may lead the acquirer into a false sense of security by allowing it to assume the systems integrator will correctly identify and meet the required reliability. Systems integration is naturally concentrated towards the end of the project when all the components to be integrated are available. Unfortunately that is also the time when cost and schedule over-runs which may have occurred in the individual components become most prominent. Most cost and schedule pressure is therefore concentrated on the systems integrator towards the end of the project and accordingly any requirements which are either not clearly specified, not measurable, or have least affect on acceptance activities are most likely to be shortcut. Reliability is one such attribute which may tend to be least visible during acceptance testing as the system generally only needs to be reliable for the limited period of that testing.

Despite the above, the act of simply employing an expert systems integrator should in itself provide some risk mitigation.

Overall, for the reasons given above the:

Service model is assessed as: Fair
5.6.3.5.2 Does this relationship model require sophisticated developers and management?

Any large information technology based system requires sophisticated developers and managers to the extent necessary to address numerous concurrent complex technical activities. The service model to some extent allows the acquirer to pass on the risks and responsibilities of the project to a specialist systems integrator and accordingly reduces the sophistication required by it. From the acquirers perspective therefore there is in theory a reduced need for sophisticated developers and managers.

Despite that, the customer naturally needs to make sufficient inquiries on a regular basis to ensure that the systems integrator has control of the project.

Overall, for the reasons given above the:

Service model is assessed as: Good
for this criteria.

5.6.3.6 Micro-agreements in the Context of Software Configuration Management

5.6.3.6.1 Does the relationship model have the capacity to enforce good software configuration management practices?

The configuration management process is an important function but conceptually reasonably straightforward. It is essentially a set of processes designed to allow changes to configured items (such as source code) in a controlled manner. Provided that the developer of a particular component follows a defined configuration management process then for that component there should not be major issues. To reduce the risk and increase confidence in the process it could be a condition of the contract that a configuration management process be in place. Standards are available which can be used as a guide and some of these are listed in Appendix A. Software Standards.

Despite the above, in this model, system wide configuration management practices are directly under the control of the systems integrator. The systems integrator itself needs to ensure it has adequate configuration management in place to manage the acquisition of sub-components and their integration into the system as a whole.

Thus in summary, the service model is considered adequate for enforcing good configuration management practices provided that adequate processes are in place and preferably required as a contractual obligation.

Overall, for the reasons given above the:

Service model is assessed as: Good
for this criteria.

5.6.3.7 Assessment of Legal Risks

What capacity does this relationship model have in minimising legal risks? Specifically, what is the likelihood of claims under contract, estoppel, trade practices or other legal doctrines if this relationship model is used?
In some respects the service model is similar to the conventional model in that the buyer forms a contract with a ‘specialist’ systems integrator who then sub-contracts with suppliers directly to obtain the components of the system for integration.

From the point of view of the buyer, one reason for entering into this type of arrangement is to pass on the risk of systems integration activities onto a specialist in the field. The question is thus whether the legal risks are minimised for the buyer by entering into a service model type of arrangement. In the previous chapter, case studies including Case Study 2 (See Section 4.6) provided an example of a situation where the Service Model was used and where a number of legal issues were examined.

In summary it is submitted that the service model does not significantly change the legal risks from those identified for the conventional model, although the burden of these risks may be spread somewhat differently. Specifically, in the conventional model, the main legal risks were identified as:

- the risk of actions for breach of contract may be increased due to the difficulty in accurately predicting the cost and schedule associated with the technical work (but see further discussion points on this under the conventional model assessment – section 5.5.3.7 above; and
- this model encourages the formation of contracts for information technology products when industry experience shows that it may be difficult to estimate the cost and schedule associated with acquiring these products. Any estimates of cost, schedule or performance of the system which therefore turn out to be incorrect, may give rise to actions in misrepresentation, and estoppel for example 592

It is submitted that in the service model the focus of these potential breaches is likely to fall on the suppliers of the components of the system as well as on the systems integrator in that those entities have contractual obligations to deliver products and services in an environment where requirements may be changing and estimating schedule, cost and quality may be difficult.

In summary it appears that the main legal risks are the same as those identified above except that there is one extra layer of subcontract to perhaps spread some of the risks further.

Overall, for the reasons given above:

The Service Model is assessed as poor in minimising legal risks particularly in relation to breach of contract actions, and to a lesser extent, in relation to misrepresentation and estoppel.

5.6.3.8 Summary of Assessment – Service Model
The following table summarises the above assessment.

<table>
<thead>
<tr>
<th>Relationship Model Evaluation Criteria</th>
<th>Effectiveness of Relationship Model in coping with micro-agreements</th>
</tr>
</thead>
</table>

592 see Case Study 1 (See Section 0 above)
<table>
<thead>
<tr>
<th>Relationship Model Evaluation Criteria</th>
<th>Effectiveness of Relationship Model in coping with micro-agreements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Micro-agreements in the context of Requirements Management</td>
<td>• Does the relationship model have a capacity to deal with inadequately defined requirements?</td>
</tr>
<tr>
<td></td>
<td>• Does the relationship model have a capacity to deal with an evolving system architecture?</td>
</tr>
<tr>
<td></td>
<td>• Does the relationship model have a capacity to deal with a system which needs to grow?</td>
</tr>
<tr>
<td></td>
<td>• Does this relationship model have the capacity to allow changes to be easily introduced into the system?</td>
</tr>
<tr>
<td>Micro-agreements in the context of Software Project Planning</td>
<td>• Does the relationship model have the capacity to deal with developments where the schedule is important?</td>
</tr>
<tr>
<td>Micro-agreements in the context of Software Tracking and Oversight</td>
<td>• Are the overheads low for this relationship model?</td>
</tr>
<tr>
<td></td>
<td>• Does this relationship model give the customer good visibility of progress?</td>
</tr>
<tr>
<td></td>
<td>• Does the relationship model give management good visibility of progress?</td>
</tr>
<tr>
<td>Micro-agreements in the context of Sub-contractor Management</td>
<td>• Does the relationship model allow the management of risks effectively?</td>
</tr>
<tr>
<td>Micro-agreements in the context of Software Quality Assurance</td>
<td>• Does the relationship model have the capacity to deal with developments where system reliability is important?</td>
</tr>
<tr>
<td></td>
<td>• Does this relationship model require sophisticated developers and management?</td>
</tr>
<tr>
<td>Micro-agreements in the context of Software Configuration Management</td>
<td>• Does the relationship model have the capacity to enforce good software configuration management practices?</td>
</tr>
<tr>
<td>Legal Risks</td>
<td>• What capacity does this relationship model have in minimising legal risks?</td>
</tr>
</tbody>
</table>

**Figure 34 - Evaluation of Service Model Attributes**
5.6.3.9 Looking at trends – Micro-agreements and the Service Relationship Model

Overall, when a micro-agreement occurs within the context of the software engineering fundamental "best practice" areas discussed above, then the following findings are made.

<table>
<thead>
<tr>
<th>Finding</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finding 9:</td>
<td>The service relationship model is likely to allow micro-agreements to occur, or at least not prevent them from occurring.</td>
</tr>
<tr>
<td>Finding 10:</td>
<td>The service relationship model is likely to provide some flexibility to enable changing circumstances to be managed effectively, specifically by either direct contract change amendments, or by appropriate software engineering processes which are contractually supported.</td>
</tr>
<tr>
<td>Finding 11:</td>
<td>The service relationship model is likely to allow the parties to a contract little discretion to accept and implement &quot;good&quot; micro-agreements (that is, those which are consistent with the overall business objectives of the parties), except by either direct contract change amendments, or by appropriate software engineering processes which are contractually supported.</td>
</tr>
<tr>
<td>Finding 12:</td>
<td>The service relationship model is likely to allow the parties to a contract the discretion to reject and not implement &quot;bad&quot; micro-agreements (that is, those which are not consistent with the overall business objectives of the parties). The main reason for this is that prima facie they are not contractually binding. Despite this, issues such estoppel, unconscionability, etc. may be relevant.</td>
</tr>
<tr>
<td>Finding 13:</td>
<td>The service relationship model is likely to result in an information technology system which is less likely to satisfy customer needs in terms of cost, quality and schedule, unless a solid software engineering process is in place, preferably as a contractual requirement on the parties.</td>
</tr>
<tr>
<td>Finding 14:</td>
<td>The legal risk of breach of contract is relatively high for the service model (compared with some other models) especially if changing requirements cannot be introduced easily into the contracting process.</td>
</tr>
</tbody>
</table>
Finding 15: The service model encourages the formation of contracts for information technology products when industry experience shows that it may be difficult to estimate the cost and schedule associated with acquiring these products. Any estimates of cost, schedule or performance of the system which therefore turn out to be incorrect, may possibly lead to actions in misrepresentation, and estoppel (for example).

Finding 16: The service relationship model may be the best choice for information technology projects where the customer recognises that significant technical risk is involved requiring the services of a specialist systems integrator. Regardless, it is considered that the best chance of achieving success with this model will occur if appropriate software engineering processes are required as contractual obligations both in the prime contract between the customer and the systems integrator, and also in subcontracts for system components.

5.6.3.10 Conclusion
The service model is one where the customer contracts with a specialist systems integrator for integration and delivery of the entire system. There are a number of advantages and disadvantages to this approach however in summary, based on information distilled from the above assessment of the model, in order to successfully implement this model:

- the systems integrator needs to fully understand the system requirements and extract these from the customer
- the systems integrator needs to clearly define a system architecture to be able to fit new components into a system
- if the requirements are fully understood and the specification is complete, then if software engineering process formalities are adhered to by the systems integrator together will all subcontractors then a high quality and reliable system should be the result
- risks are taken on by the systems integrator to a large degree although the customer presumably will need to pay extra to the systems integrator to take on this risk
- the model doesn't easily allow for corrections half way through unless either contract amendments are used, or solid software engineering processes are in place and enforceable
- if schedule is important then this model can move the risk of schedule delays onto the systems integrator as an expert in the field

Of fundamental importance to the successful use of this model is the necessity for the systems integrator to fully understand and document the requirements. Any uncertainty in specifications provided to sub-contractors will increase the risk of components not interacting together successfully. Specifically, the systems integrator needs an effective requirements management process in place to achieve this aim, and that process ideally should be required as a contractual obligation. A mechanism for coping with changing
requirements is essential, and an effective software engineering process can help in this regard.

Overall, this model has the advantage of being relatively simple to understand, as well as providing some contractual certainty over cost, schedule and quality and giving the customer some opportunity to reduce its risk. On the negative side however, it requires that the systems integrator has an in-depth knowledge of system requirements and technical architectural issues, and also that any errors or omissions in sub-contracted specifications may be difficult to rectify by the systems integrator without the overheads of specific contractual amendments.

5.7 CAPTIVE MODEL

5.7.1 Description
The captive relationship model is one where an attempt is made to provide an incentive to both the acquirer and the developers to work together to achieve common objectives. Specifically, a separate company is formed by both the acquirer and the supplier to provide products and services to the acquirer.\(^\text{593}\)

5.7.2 Example
Based on the aircraft training simulator example used to illustrate the previous model, if a customer required the development of a flight simulator for training crews, then it could possibly use the captive model to acquire the system. A systems integrator would be identified and a separate company formed by the systems integrator and the customer. The objects of this new corporation would be to provide products and services to the customer as required. The company would be responsible for systems integration work and may sub-contract components of the system to third parties as it sees fit. As an example of some of the work which may be undertaken for an aircraft simulator, the system may require, for example, integration of two different navigation sub-systems (for redundancy) such as a satellite based Global Positioning System (GPS) sensor together with an Inertial Navigation System (INS). As these two sources of navigation data need to be integrated into the flight simulation model the types of data, data rates etc. need to be consistent. In this situation, the new company could contract with the suppliers of each of the two navigation systems directly to provide the required hardware and software, and once fully integrated by the company, would be delivered to the customer.

5.7.3 Analysing the Relationship Model

The primary purpose of this relationship model is to gain a greater degree of co-operation between the parties and to ensure that there is a common objective. Having a separate corporation formed for the purpose of delivering services and products to the customer provides one way of doing this. The objectives of the corporation are defined when the corporation is established. Because both the supplier and the buyer of products and services have ownership of the new corporation then both parties have a say in running the organisation. The directors of the corporation have legally enforceable rights and obligations towards various parties to ensure that the corporation is managed in a way which meets its objectives.

Presumably also there is at least some degree of competition introduced into the arena because the buyer is still able to obtain products and services from other parties as required despite its interest in the new corporation.

5.7.3.1 Micro-agreements in the context of Requirements Management

5.7.3.1.1 What capacity does this relationship model have in dealing with inadequately defined requirements?

The issue of requirements management has been discussed above in reasonable detail (e.g. see Sections 2.5, 0) however the issue was summarised in one article as:

“One of the most common reasons for failure is not the IT itself, it is the management of the project, specifically failure to keep a tight rein on the cost and scope of a project.”

Banaghan reports on a proposed solution to the problem being to avoid the rigid specification of requirements at the start, but rather the forming of an ongoing relationship aimed at evolving the requirements over time:

“So rather than us [the buyer] giving them the specifications of what we wanted, we worked with them [the suppliers] so they understood from a functional sense what our requirements are, and they built the scope around it.”

In one sense the captive relationship model allows this type of interaction to take place more easily than through some of the more traditional relationship models described above (that is the Conventional Model and the Service Model). In essence this is the case because in the different relationship models, the objectives of the parties involved in the development is different. In the Conventional Model the objective of each of the subcontractors or service providers is likely to be to meet its contractual obligations related to an individual component of the system. If a change in the requirements is identified which affects the system as a whole, then that in itself does not change the objectives of a subcontractor which already has defined contractual obligations to meet.

Alternatively in the Captive Model (and as we will see in the next model also - the Joint Venture Change Agent Model), the objectives of the parties are likely to have a different focus. Specifically, rather than having an objective of building an information technology system to a rigid specification, the objective of the corporation formed by the buyer and the suppliers may be broader in scope. For example, the overall objective may be to deliver a new information technology system to the customer which has a measurable

impact on the efficiency of the customers business to the extent that it is capable of supporting business growth of 20% per annum and achieves a minimum return on investment over a three year period.

Because of the dynamics of information technology it is likely that many things will change over the period of time required to develop and install the system. The Captive Model allows the parties to focus on the "big picture" objectives and to be flexible and adaptive to this change rather than getting bogged down with rigid specifications unnecessarily.

Of course it is also true that in order to be able to develop an information technology system the developers must know in fine detail what the system is required to do. The developers therefore are likely to need detailed specifications to work with however with the Captive Model there is some degree of flexibility to enable changes to the specifications to be made to meet global objectives without forcing changes to contractually binding obligations.

On the negative side however it is also possible that because a flexible relationship has been established, continued changes to requirements are so disruptive to the development efforts that progress may not be a good as initially anticipated. Thus the scope of the project, or in other words the system requirements may continue to change at a rate which became unmanageable:

"What often causes a cost blow-out in large-scale IT projects is that vendors are asked to customise software to replicate a company's idiosyncratic systems."  

So if the customer is not bound to accept a system with rigidly defined requirements, additional or changing functionality could cause cost and schedule increases if not managed appropriately.

Overall however, for the reasons given above the:

Captive model is assessed as: Good
for this criteria.

5.7.3.1.2 What capacity does this relationship model have in dealing with an evolving system architecture?

If a change is identified which requires architectural modifications then the Captive Relationship Model has some advantages over other relationship models.

Overall architectural issues are likely to be under the control of the systems integrator and with the Captive Model it is likely that a new corporation (consisting of both customer and suppliers) will take on this role. To some extent then the fact that architectural changes are needed is under the control of that corporation, and as representatives of all parties are presumably on the board of such a corporation then the decisions made will be based on meeting the overall company objectives rather than meeting detailed technical or contractual requirements.

It was discussed briefly in the previous Section how the corporation objectives (at least using a Captive relationship Model) are likely to be worded in a general sense to enable the achievement of "big picture" business objectives. Accordingly the management of technical details is likely to be relatively straightforward from a legal perspective. Simply

---

597 Remember that the "architecture" of an information technology system describes how the components of that system fit together to meet the specified requirements.
put, if the architectural changes are in line with corporation objectives then the new corporation itself can internally make the decision to implement them or not.

The key factor here is that decisions are made internally within one legal entity and thus do not cross the boundary of any legal relationship. The new corporation, as system integrator, makes the decision to implement the architectural changes or not.

In practice the customer will have some contract with the systems integrator but it is likely that this contract will focus on the business relationship rather than the specific product to be developed. The systems integrator in that case would have the responsibility of ensuring overall objectives are met.

Overall, for the reasons given above the:

Captive model is assessed as: Good
for this criteria.

5.7.3.1.3 What capacity does this relationship model have in dealing with a system which needs to grow?

The previous question addressed the issue of changing requirements during the period when it is being developed. This question looks at the issue of once the system is operational, how easy is it to upgrade its capacity and capability to cope with changing business needs over time.

The Captive Model provides for the establishment of a separate corporation consisting of members of the customer and the suppliers collectively. As the result is a separate legal entity in some ways this model is better adapted for the ongoing provision of services rather than for a single project such as the development of a single information technology system. Thus this model may be suitable for providing a continuous service associated with maintenance of an existing system.

In terms of providing support for dealing with a system that needs to grow, it is considered that certainly longer term issues such as system growth are more likely to be addressed under the Captive Model than possibly some of the other models. A legal entity established for the sole purpose of meeting co-ordinated objectives of the customer and supplier is likely to consider longer term maintenance. A conventional development model on the other hand is likely to result in work focused on meeting specific contractual objectives identified at the time of contracting. Of course it depends on the obligations which are specified in the contract but if there is a some incentive for achieving a long term business relationship then naturally those longer term business needs are more likely to be met in a changing world.

"Open system specification" techniques may also be useful in this regard because such systems are designed in a manner which allows them to grow and adapt while at the same time interconnect with other systems with minimal disruption.

On the negative side the captive model may result in a long term relationship which has the effect of reducing competition (by effectively allowing the customer to obtain products and services from one entity – the Captive corporation), and thus actually increasing costs. Of course the customer is likely to share in profits of the new corporation but that issue aside, the system and its continued development and maintenance may be placed under the control of a single corporation. If that corporation is commercially tied to any

---

598 an "open system" is one which demonstrated quality attributes suitable for connection with other systems. In particular, such quality attributes include: interoperability, scalability, and portability.
particular products or services, which is likely to be the case if the original suppliers are directors of the corporation, then that may further limit the long term options available to the customer for enhancing and maintaining the system.

Overall, for the reasons given above the:

| Captive model is assessed as: | Fair to Good for this criteria. |

5.7.3.2 Micro-agreements in the Context of Software Project Planning

5.7.3.2.1 Does the relationship model have the capacity to deal with developments where the schedule is important?

The Captive Model may be useful in achieving delivery within a rigidly defined timeframe however it does have some drawbacks.

On the one hand, a conventional model (for example) is one where contractual issues related to the *product* are effectively paramount to the parties. Contracts are put in place between the systems integrator and sub-contractors which ideally clearly and fully specify the requirements of the system, and also ideally specify clear schedules and milestones. Thus these types of models have a contractually enforceable schedule which, if not met, may provide legal rights and remedies to the other parties.

In the Captive Model on the other hand, the focus of the contract is likely to be on the *relationship* rather than the *product*. The responsibility for the work is likely to vest in a single legal entity – a new corporation formed by the customer and suppliers collectively. The corporation cannot have a legal contractual right against itself if it does not achieve target schedules. It is likely however that the corporation will have a contract between itself and the customer to perform the work required. As the customer is also on the board of the corporation however it may present a situation where the strict contractual rights in relation to schedules may not be fully in place due to an understanding that the new corporation has been specifically set up to meet the needs of the customer and supplier collectively.

Despite this however it may also be the case that a Captive approach gives management of the project the flexibility to prioritise system requirement and make decisions to reduce functionality in order to achieve delivery of the system in the required timeframe.

Overall, for the reasons given above the:

| Captive model is assessed as: | Fair to Good for this criteria. |

5.7.3.3 Micro-agreements in the Context of Software Project Tracking and Oversight

5.7.3.3.1 Are the overheads low for this relationship model?

This is a sophisticated model in that a new corporation is set up to provide products and services to the customer. From that perspective, the overheads of setting up a corporation and managing its day to day operations as well as meeting regulatory requirements may be relatively higher than some of the relationship models previously discussed.

Overall, for the reasons given above the:

| Captive model is assessed as: | Poor (i.e. high overheads) for this criteria. |
5.7.3.3.2 Does this relationship model give the customer good visibility of progress?

Because the customer in this model is a director of the corporation performing the work then it has very good visibility of progress. It also has a good ability to influence decisions about the products and services to be delivered. As the customer is a director of the corporation performing the work, it has in fact legal obligations for ensuring good management of the corporation.

Overall, for the reasons given above the:

Captive model is assessed as: Very Good
for this criteria.

5.7.3.3.3 Does the relationship model give management good visibility of progress?

Under the Captive Model, management responsibility for delivery of products and services is likely to fall on the new corporation. In that sense, the captive model does give good visibility through the fact that the same corporation which is managing the work is also likely to be supplying the products and services. As there are no legal boundaries separating management and developers then in theory the flow of management information is unlikely to be stifled by contractual, intellectual property or other such legal issues. Provided that there is effective management within the new corporation then solid visibility of progress should be available.

Overall, for the reasons given above the:

Captive model is assessed as: Very Good
for this criteria.

5.7.3.4 Micro-agreements in the Context of Sub-Contractor Management

5.7.3.4.1 Does the relationship model allow the management of risks effectively?

Once again, this relationship model is designed to bring the suppliers of the system and the customer into a co-operative team environment. The fact that legal boundaries between the parties have been to some degree eliminated (by having customer and supplier representatives in a single legal entity) allows for an easier flow of information needed to assess risk.

In the conventional and service model for example, the customer is likely to only obtain data as specified in the particular contracts and sub-contracts such as, for example, attendance at design reviews and access to preliminary documentation. For contractual and intellectual property reasons however some restrictions on accessing information needed to fully assess project risks may not be available.

On the other hand a traditional service model type of arrangement may enforce strict risk management practices and accounting standards while a new corporation set up using the Captive Relationship model may not. Specific situations may result in one model being better than another however overall it is assessed that the Captive Model has the ability to give good visibility of information required to manage risks effectively.

599 Various standards for Cost Schedule Control Systems are in existence. The United States Department of Defence Instruction 5000.2 is one example.
Overall, for the reasons given above the:

Captive model is assessed as: Good
for this criteria.

5.7.3.5 Micro-agreements in the Context of Software Quality Assurance

5.7.3.5.1 Does the relationship model have the capacity to deal with developments where system reliability is important?

System reliability is one quality assurance issue which may be important to the customer. The Captive Model is unlikely to inherently provide any additional support initially to ensure that it is addressed adequately however it does provide a degree of continuity required to make continuous improvements to its development processes if necessary.

Quality processes are usually developed over time by organisations depending upon specific needs. Some organisations may "tune" their processes to achieve fast turnaround or low cost items at the expense of other quality attributes such as reliability. The fact that in a Captive Model both the customer and the supplier are represented in the same organisation allows the "culture" of each individual organisation to be merged over time. Thus if the suppliers culture is "tuned" to rapid development while the customers culture is more attuned to reliability then the culture which develops within the new organisation will hopefully meet both needs. Because of this factor it is considered that the Captive model gives a good opportunity for achieving a "quality" product.

On the negative side however it is important to realise that management of the new corporation needs to be effective enough to recognise the issues associated with quality and to put in place appropriate processes. In the more conventional models it is likely that, at least for the development of larger systems, particular quality standards will be mandated in the contract between the customer and the systems integrator. In the Captive Model there may not be such a legally enforceable contractual requirement although it certainly could be part of the overall corporation objectives to provide products and services at the required quality.

Overall, for the reasons given above the:

Captive model is assessed as: Fair to Good
for this criteria.

5.7.3.5.2 Does this relationship model require sophisticated developers and management?

The Captive Model aims to delegate the responsibility of development and management decisions to a separate corporation formed for that purpose. To that extent the customer is relieved of having to use sophisticated techniques to either manage or develop a system, but rather can (in theory) give broad direction to the systems integrator (that is, the separate corporation). The direction given could, for example, take the form of advice on the customer's business needs both present and future. From that perspective the customer can off-load as much 'sophistication' and responsibility to the separate corporation as it sees fit.

It must also be remembered that the customer is also a significant part of the new corporation formed. Accordingly if that corporation has a heavy responsibility for the delivery of a system then by necessity that corporation is likely to needs sophistication in both its managerial approach as well as technical capabilities. Of course every project is different however as we have previously seen the overheads for this model are relatively
high compared to other models and as such the captive model is more likely to be used for larger more complex projects. Large complex projects can be risky (as discussed in Chapter 2 above) and accordingly need significant sophistication to achieve desired outcomes.

Another key point to remember is that, despite the close relationship and heavy involvement by the customer in the corporation set up to perform the work, it is that corporation as a separate legal entity which bears the responsibility for delivering the system. As a separate entity it has its own objectives to achieve and legal responsibilities in doing so. The benefit of the captive relationship model is however that communication between customer and suppliers can be more dynamic and flexible in accommodating changing requirements when compared to some other relationship models such as those based on a single rigid contract and specification.

Thus with the captive model the customer does not particularly need sophisticated developers and managers as that expertise is provided by a separate corporation set up for that purpose. That separate corporation does however need experience and processes in place suitable for the work at hand. In large or complex information technology projects the overall sophistication may be considerable for the separate corporation.

Overall, for the reasons given above the:

Captive model is assessed as: Good
for this criteria.

5.7.3.6 Micro-agreements in the Context of Software Configuration Management

5.7.3.6.1 Does the relationship model have the capacity to enforce good software configuration management practices?

It has been previously that the configuration management process is an important one but conceptually reasonably straightforward. It is essentially a set of processes designed to allow changes to configured items (such as source code) in a controlled manner. The main issue then is ensuring that the developers and system integrator follow a defined process and strictly adhere to it to the extent necessary to ensure changes to parts of the system are tracked effectively. This can usually be enforced contractually by requiring the developers to meet specified standards such as some of those identified in Appendix A, Software Standards.

In the captive model there is some risk that the customer will rely to a large extent on the separate corporation formed to perform the work. As the customer is a member of that separate corporation then it may assume that it does not need detailed contracts between itself and that corporation because it has a share in the control of that corporation in meeting its objectives.

To avoid this risk either the new corporation itself needs to ensure it has adequate configuration management in place to manage the acquisition of sub-components and their integration into the system as a whole, or otherwise the customer needs to contractually ensure that this is the case.

Thus in summary, the captive model is adequate for enforcing good configuration management practices provided that processes are in place and preferably required as a contractual obligation.
Overall, for the reasons given above the:

Captive model is assessed as: Fair to Good
for this criteria.

5.7.3.7 Assessment of Legal Risks
What capacity does this relationship model have in minimising legal risks? Specifically, what is the likelihood of claims under contract, estoppel, trade practices or other legal doctrines if this relationship model is used?

In the captive model a separate company is formed by both the acquirer and the supplier to provide products and services to the acquirer. In summary, the main reason for implementing this relationship model is to gain a greater degree of co-operation between the parties and to ensure that there is a common objective. Because both the supplier and the buyer of products and services have ownership of the new corporation then both parties have a say in running the organisation and have an interest in resolving issues.

The key factor here is that decisions are made internally within one legal entity and thus do not cross the boundary of any legal relationship. The new corporation, as system integrator, makes decisions on implementation of the system internally within the corporation and accordingly, the legal risk associated with interacting with separate entities is reduced. In relation to specific legal risks the following points are made:

- In the captive model the risk of breach of contract between the supplier and customer is minimised because they become part of the same legal entity. Because the supplier and customer in this model are also directors of the corporation supplying the system, they each have significant managerial influence and control over the activities of the corporation to the extent needed to achieve its objectives.
- In the captive model there are few legal boundaries separating management and developers and accordingly the flow of management information is less likely to be stifled by contractual, intellectual property or other such legal issues.
- With the captive model, both the supplier and customer are directors of the captive corporation and accordingly have obligations as directors. One potential legal risk is that if either party acts in its own self-interest rather than in the interest of the captive corporation then directors duties (including possibly fiduciary obligations) may be breached.

Overall, for the reasons given above:

The Captive Model is assessed as good in minimising legal risks particularly in relation to breach of contract actions, although it may slightly increase the risk of breaches of the Corporations Law in respect of director's duties.

5.7.3.8 Summary of Assessment – Captive Model
The following table summarises the above assessment.

<table>
<thead>
<tr>
<th>Relationship Model Evaluation Criteria</th>
<th>Effectiveness of Relationship Model in coping with micro-agreements</th>
</tr>
</thead>
</table>

Paul P. Parnell
<table>
<thead>
<tr>
<th>Relationship Model Evaluation Criteria</th>
<th>Effectiveness of Relationship Model in coping with micro-agreements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Micro-agreements in the context of Requirements Management</td>
<td>Does the relationship model have a capacity to deal with inadequately defined requirements?</td>
</tr>
<tr>
<td>Micro-agreements in the context of Requirements Management</td>
<td>Good (See 5.7.3.1.1)</td>
</tr>
<tr>
<td>Micro-agreements in the context of Requirements Management</td>
<td>Does the relationship model have a capacity to deal with an evolving system architecture?</td>
</tr>
<tr>
<td>Micro-agreements in the context of Requirements Management</td>
<td>Good (See 5.7.3.1.2)</td>
</tr>
<tr>
<td>Micro-agreements in the context of Requirements Management</td>
<td>Does the relationship model have a capacity to deal with a system which needs to grow?</td>
</tr>
<tr>
<td>Micro-agreements in the context of Requirements Management</td>
<td>Fair to Good (See 5.7.3.1.3)</td>
</tr>
<tr>
<td>Micro-agreements in the context of Requirements Management</td>
<td>Does this relationship model have the capacity to allow changes to be easily introduced into the system?</td>
</tr>
<tr>
<td>Micro-agreements in the context of Requirements Management</td>
<td>Not assessed as covered by previous questions</td>
</tr>
<tr>
<td>Micro-agreements in the context of Software Project Planning</td>
<td>Does the relationship model have the capacity to deal with developments where the schedule is important?</td>
</tr>
<tr>
<td>Micro-agreements in the context of Software Project Planning</td>
<td>Fair to Good (See 5.7.3.2.1)</td>
</tr>
<tr>
<td>Micro-agreements in the context of Software Tracking and Oversight</td>
<td>Are the overheads low for this relationship model?</td>
</tr>
<tr>
<td>Micro-agreements in the context of Software Tracking and Oversight</td>
<td>Poor (relatively high overheads) (See 5.7.3.3.1)</td>
</tr>
<tr>
<td>Micro-agreements in the context of Software Tracking and Oversight</td>
<td>Does this relationship model give the customer good visibility of progress?</td>
</tr>
<tr>
<td>Micro-agreements in the context of Software Tracking and Oversight</td>
<td>Very Good (See 5.7.3.3.2)</td>
</tr>
<tr>
<td>Micro-agreements in the context of Software Tracking and Oversight</td>
<td>Does the relationship model give management good visibility of progress?</td>
</tr>
<tr>
<td>Micro-agreements in the context of Software Tracking and Oversight</td>
<td>Very Good (See 5.7.3.3.3)</td>
</tr>
<tr>
<td>Micro-agreements in the context of Sub-contractor Management</td>
<td>Does the relationship model allow the management of risks effectively?</td>
</tr>
<tr>
<td>Micro-agreements in the context of Sub-contractor Management</td>
<td>Good (See 5.7.3.4.1)</td>
</tr>
<tr>
<td>Micro-agreements in the context of Software Quality Assurance</td>
<td>Does the relationship model have the capacity to deal with developments where system reliability is important?</td>
</tr>
<tr>
<td>Micro-agreements in the context of Software Quality Assurance</td>
<td>Fair to Good (See 5.7.3.5.1)</td>
</tr>
<tr>
<td>Micro-agreements in the context of Software Quality Assurance</td>
<td>Does this relationship model require sophisticated developers and management?</td>
</tr>
<tr>
<td>Micro-agreements in the context of Software Quality Assurance</td>
<td>Good (See 5.7.3.5.2)</td>
</tr>
<tr>
<td>Micro-agreements in the context of Software Configuration Management</td>
<td>Does the relationship model have the capacity to enforce good software configuration management practices?</td>
</tr>
<tr>
<td>Micro-agreements in the context of Software Configuration Management</td>
<td>Fair to Good (See 5.7.3.6.1)</td>
</tr>
<tr>
<td>Legal Risks</td>
<td>What capacity does this relationship model have in minimising legal risks?</td>
</tr>
<tr>
<td>Legal Risks</td>
<td>Good (See 5.7.3.7)</td>
</tr>
</tbody>
</table>

**Figure 36 - Evaluation of Captive Model Attributes**

Paul P. Parnell
5.7.3.9 Looking at trends – Micro-agreements and the Captive Relationship Model

Overall, when a micro-agreement occurs within the context of the software engineering fundamental "best practice" areas discussed above, then the following findings are made:

Finding 17: The captive relationship model is likely to allow micro-agreements to occur, and in fact encourage them because there are no legal or contractual boundaries as such between parties who form part of the captive corporation.

Finding 18: The captive relationship model is likely to provide a good degree of flexibility to enable changing circumstances to be managed effectively, through appropriate changes to the software project being dictated by the corporation management itself, of which interested parties are represented. In addition, appropriate software engineering processes can also be put in place.

Finding 19: The captive relationship model is likely to allow interested parties a large amount of discretion in whether it accepts and implements "good" micro-agreements (that is, those which are consistent with the overall business objectives of the captive corporation).

Finding 20: The captive relationship model is likely to allow interested parties the discretion to reject and not implement "bad" micro-agreements (that is, those which are not consistent with the overall business objectives of the parties).

Finding 21: The captive relationship model is likely to result in an information technology system which is more likely to satisfy customer needs in terms of cost, quality and schedule, provided that firstly, the additional overheads of setting up and managing a separate corporation to perform the work is justified. Large or complex software based projects may provide this justification. Secondly, a solid software engineering process is in place for managing the complex issues which are likely to arise in larger projects.

Finding 22: In the captive model the risk of breach of contract between the supplier and customer is minimised because they become part of the same legal entity. Because the supplier and customer in this model are also directors of the corporation supplying the system, they each have significant managerial influence and control over the activities of the corporation to the extent needed to achieve its objectives.
Finding 23: With the captive model, both the supplier and customer are directors of the captive corporation and accordingly have obligations as directors. One potential legal risk is that if either party acts in its own self-interest rather than in the interest of the captive corporation then directors' duties (including possibly fiduciary obligations) may be breached.

Finding 24: The captive relationship model may be a good choice for information technology developments where the customer recognises that significant technical risk is involved and where business needs and hence system requirements are either not fully known, or are likely to change significantly over the life of a project. On the other hand, smaller well defined projects may not require the complexity and cost of setting up and running a separate corporation to perform the work. Despite the above points, it is considered that the best chance of achieving success with this model will occur if appropriate software engineering processes are put in place.

5.7.3.10 Conclusion

The Captive model is one where the customer and suppliers form a separate corporation to provide products and services to the customer. There are a number of advantages and disadvantages to this approach however in summary, based on information distilled from the above assessment of the model, in order to successfully implement this model:

- the objectives of the captive corporation must be consistent with the business needs of both the customer and suppliers
- interested parties need to have some representation on the board of the captive corporation so that specific interests are addressed adequately
- a systems engineering process should be set up within the captive organisation to enable complex project issues to be managed
  - risks are taken on by captive corporation to a large degree rather than the individual suppliers of components (if those suppliers are part of the captive corporation)
  - the model does allow for changes to be made to the system throughout the life of the development and integration as the decision on whether to implement a change is made by a single entity – the captive corporation

Of fundamental importance to the successful use of this model is the necessity for the corporate objectives to be identified correctly. Once identified then the captive relationship model should provide flexibility to accommodate change and manage risk, provided that the change is consistent with the objectives of the captive corporation.

Overall, this model has the advantage of giving a large amount of flexibility to the parties to accommodate changing circumstances. It creates a single legal entity which eliminates legal boundaries and lets decision makers operate within the one organisation. Micro-agreements which occur within the context of this model can be implemented or rejected easily depending upon the overall assessment of the captive corporation members.

On the other hand however, the freedom to allow changes to the system easily, or at least more easily than in a conventional model may lead to "requirements creep". Requirements creep occurs when changes to the requirements change at a rate faster than that which
the developers can implement. A solid systems engineering process can help to manage this and other risks.
5.8 JOINT VENTURE CHANGE AGENT MODEL

5.8.1 Description
The Joint Venture Change Agent relationship model is one where an attempt is made to provide an incentive to both the acquirer of the system and the developers to work together to achieve common objectives. Specifically, a joint venture is formed by both the customer and the supplier to provide products and services to the customer, and also to provide products and services to third party customers on a commercial basis. This is similar to the captive model described above except that with the Joint Venture Change Agent Model the products and services are also sold to third party buyers.\(^{600}\)

5.8.2 Example
Based on the aircraft training simulator example which was previously used to illustrate the other models, if a customer required the development of a flight simulator for training crews, then it could use the Joint Venture Change Agent Model to acquire the system. A systems integrator would be identified and a joint venture formed by the systems integrator and the customer. The objective of the Joint Venture would be to provide products and services to the customer as required, and in addition, also provide products and services to third party customers as well. The parties to the Joint Venture would be responsible for systems integration work and may sub-contract components of the system to third parties as they see fit. As an example of some of the work which may be undertaken for an aircraft simulator, the system may require integration of two different navigation sub-systems (for redundancy) such as a satellite based Global Positioning System (GPS) sensor together with an Inertial Navigation System (INS). As these two sources of navigation data need to be integrated into the flight simulation model the types of data, data rates etc. need to be consistent. In this situation, the Joint Venture could, for example, contract with the suppliers of each of the two navigation systems directly to provide the required hardware and software, and once fully integrated by the company, would be delivered to the customer.

5.8.3 Analysing the Relationship Model
The primary purpose of this relationship model is to gain a greater degree of co-operation between the parties and to ensure that there is a common objective. Having a separate joint venture formed for the purpose of delivering products and services to the customer provides one way of doing this. The ability to supply products and services to third party customers would be another benefit.

---
customers also provides a way of stabilising the business cash flow through additional sales. The objectives of the joint venture are defined when the joint venture is established. Because both the supplier and the buyer of products and services have an interest in the new joint venture then both parties have a say in the management of activities associated with the venture.

5.8.3.1 Micro-agreements in the context of Requirements Management

5.8.3.1.1 What capacity does this relationship model have in dealing with inadequately defined requirements?

The issue of requirements management has been discussed above in reasonable detail (for example see Sections 2.5, 0). Essentially the issue comes down to ensuring that the product delivered is appropriate in terms of its cost, quality and delivery timeframe. In legal terms this could translate to "fitness for purpose" and "merchantability" type requirements called for in the Sales of Goods Acts (for example). So what support does this relationship model give to the management of requirements?

A brief discussion of joint ventures was provided earlier in this chapter and one example of a definition was provided. The relevant parts of that definition for the present purpose is that the parties to the joint venture "... agree by contract to engage in some common, usually ad hoc undertaking for joint profit by combining their respective resources...

Even though the basis of a joint venture is a contractual arrangement between the parties, it can provide an additional set of advantages which may not be immediately obvious. Traditionally joint ventures have been used in commercial ventures involving high risk activities such as mining exploration. From a purely contractual perspective it would be possible for one company to contract with another to find (for example) gold deposits in return for a set fee. Because of the risks involved however it is unlikely that any company would agree to such a contract because the risk of breach of contract for non-performance of its obligations to find gold would be too high. Rather, a joint venture arrangement allows a combining of resources of all parties to achieve a common objective. Usually with such a joint venture arrangement there is a mutual sharing or risk and reward built into the arrangement and the contractual obligations support the achievement of a common objective (by for example, placing contractual obligations on a party to use its best endeavours).

In addition to this contractual perspective, the joint venture arrangement itself may bring fiduciary obligations into play which are unlikely to be available in some of the more traditional (or purely contractual) models.

The important point in respect of the Joint Venture Change Agent model for information technology developments is that it allows the combining of resources of all parties to achieve a common objective, and imposes fiduciary as well as contractual obligations on the parties to do so.

How then does this support a requirements management activity? Simply put, if the common objective of a joint venture is to produce (say) an integrated computer system for the customer, then if it is understood that the requirements management activity can a relatively high risk part of the process (refer to Section 2.5 for further discussion on this issue), a joint venture arrangement may provide the flexibility needed for a successful outcome. More specifically, the parties to the joint venture may have contractual as well as a fiduciary obligations to ensure that the requirements management activities are

performed adequately (depending of course on the joint venture agreement). Historically, in a large complex system it is unlikely that all the requirements will be identified at the outset except in broad terms and accordingly some way of managing new requirements as they are identified needs to be established.

Having a joint venture type of arrangement places the focus on the overall objectives of the venture and gives the parties the method (and indeed the legal obligation through contract and equitable doctrines such as fiduciary obligations) to ensure those objectives are met. In a purely contractual arrangement (that is, where there is no joint venture relationship as such) it is likely that the parties will be naturally focused on meeting their respective contractual obligations – whether or not those obligations are consistent with the ultimate objectives of the parties. As an example, the specific contractual obligations may be related to delivering a system as specified in a requirements document, although if the requirements of the system change then the system eventually delivered may not entirely meet the business needs of the customer. Traditionally one way of coping with this situation is to specify the system requirements as generally as possible so as to allow some scope for implementing the system in different ways. This method has problems itself however as it can lead to some degree of uncertainty.

At the time a contract comes into existence it will more likely than not be consistent with the overall objectives of the parties however as we have seen from a previous chapter the requirements management activities associated with large complex information technology projects are likely to involve considerable change. A joint venture type arrangement can help cope with the management of this change through providing the parties with sufficient flexibility to make decisions consistent with the overall objectives of the venture, while at the same time being relatively free from rigid contractual conditions which may, over time, become less consistent with those objectives. In essence, the contractual focus is on the relationship rather than the specific product being developed. At the same time the "safety nets" of firstly, a legally enforceable fiduciary relationship which imposes significant obligations on each party to carry out its activities in an appropriate manner, and secondly, the fact that individually joint venturers are not responsible for the acts of co-venturers provides a good balance for the management of requirements.

Because of the dynamics of information technology it is likely that many things will change over the period of time required to develop and install the system. The Joint Venture Change Agent Model allows the parties to focus on the "big picture" objectives and to be flexible and adaptive to this change rather than getting bogged down with rigid specifications and contractual obligations unnecessarily. Naturally specifications for the system still need to be developed to provide some concrete certainty to the developers however the joint venture model allows change to occur within the appropriate context.

On the negative side however, it is also possible that because a flexible relationship has been established, continued changes to requirements are so disruptive to the development efforts that progress may not be a good as initially anticipated. Thus the scope of the project, or in other words the system requirements may continue to change at a rate which became unmanageable. "Requirements creep" is one way in which this type of change has been described and it needs to be managed appropriately to avoid problems.

Overall however if the changes are consistent with the objectives of the joint venture in terms of quality, cost and schedule then it is suggested that this is likely to be less of a problem than if no flexibility to modify the requirements exists.

---

Overall however, for the reasons given above the:

Joint Venture Change Agent model is assessed as: Very Good
for this criteria.

5.8.3.1.2 What capacity does this relationship model have in dealing with an evolving system architecture?

If a change is identified which requires architectural modifications then the Joint Venture Change Agent Relationship Model has some advantages over some of the more conventional models discussed above.

Overall, architectural issues are likely to be under the control of the systems integrator and with the Joint Venture Change Agent Model it is likely that one of the parties to the joint venture will take on this role. Accordingly, architectural changes will be made presumably with the objectives of the joint venture in mind, and specifically in accord with the legal obligations of the joint venturer.

It was discussed briefly in the previous Section how the joint venture objectives are likely to be worded in a general sense to enable the achievement of overall business objectives. Accordingly the management of technical details is likely to be relatively straightforward from a legal perspective. Simply put, if the architectural changes are in line with joint venture objectives then the members of the joint venture itself can collectively (within the limitations of the joint venture agreement) make the decision on whether to implement the changes or not.

In the relationship model discussed previously (the Captive Model), the key factor was that such decisions were made internally within one legal entity (a new corporation formed by the customer and the supplier) and thus did not cross the boundary of any legal relationship. A joint venture is not a separate legal entity and as such the advantages are not so. In the captive model, a single legal entity is likely to be responsible for making architectural decisions related to systems integration work. In the Joint Venture Change Agent Model however there is more than one entity involved (e.g. customer and systems integrator) however they are bound by fiduciary obligations to act in achieving the collective objectives of the joint venture. In practice it is likely that one party to the joint venture will have greater expertise and resources to undertake the detailed systems integration work. If that party is however bound by contractual and fiduciary obligations within the context of specific joint venture objectives then in theory decisions on the system architecture should be within the expectations of the other parties to the venture.

Overall, for the reasons given above the:

Joint Venture Change Agent model is assessed as: Good
for this criteria.

5.8.3.1.3 What capacity does this relationship model have in dealing with a system which needs to grow?

The previous questions addressed the issue of change during the period of development and integration – that is, before the system is released. This question is intended to look at the issue of change after the system has been delivered, that is, in the maintenance phase of the system.

---

603 the “architecture” of an information technology system describes how the components of that system fit together to meet the specified requirements.
We saw in the previous relationship model, (the Captive Model) that it allowed for the establishment of a separate corporation consisting of members of interested parties such as the customer and the supplier(s). As the result is a separate legal entity, in some ways that model is better adapted for the ongoing provision of services rather than for a single project such as the development of a single information technology system. The reason for this is that a corporate infrastructure, one put in place, is suited to long term relationships if required.

A joint venture arrangement, on the other hand is traditionally put in place to achieve a specific objective rather than a continuous ongoing relationship. In that sense, long term maintenance and support of a delivered system may not be consistent with the a legal and business "infrastructure" which has been "tuned" for the achievement of a specific objective such as the delivery of an integrated information technology system.

Despite this, naturally if the objectives of the joint venture are defined to include the long term maintenance and support of a system then there is no reason why the Joint Venture Change Agent relationship should, in itself, cause problems. Indeed having the flexibility to make decisions consistent with changing business needs is likely to be somewhat preferable if (for example) both the customer and systems integrator have an incentive to achieve common objectives.

Traditionally, software maintenance contracts are based on "time and materials" type arrangements which allow the actual costs of changes to the system to be billed as they are incurred. It is also common for performance goals to be specified in a maintenance contract and as one example, it could take the form of (say) "less than 2 hours down time per month". If a joint venture arrangement does not have these types of goals explicitly stated then there may be some uncertainty as to the exact responsibilities of the parties.

Thus this model may be suitable for providing a service associated with continuous maintenance of an existing system, although it is suspected that some of the other models may prove to be more appropriate in this regard unless there is a very clear focus of the joint venture parties on what exactly their obligations are with respect to maintenance activities.

As a separate but related issue, when a system is undergoing its initial development, certain engineering and architectural decisions can affect the ease and cost of long term maintainability and supportability. In terms of providing support for dealing with a system that needs to grow, it is considered that certainly longer term issues such as system growth are more likely to be addressed under a model such as the Joint Venture Change Agent Model than possibly some of the other models discussed earlier (such as the Conventional model) provided that this aim is consistent with the overall objectives of the joint venture. In other words if the joint venturers set out to produce a system which suits business needs for a long period of time, additional effort may be made to ensure the system is maintainable.

A more conventional relationship model on the other hand may be more likely to result in work focused on meeting specific contractual objectives. Of course it depends on the obligations which are specified in the contract but if there is a some incentive for delivering a product which is easier and cheaper to maintain and adapt to long term business needs then so much the better. A joint venture arrangement can be set up to provide the flexibility necessary to adapt the system to changing business needs, and provide the system with sufficient maintainability to be able to adapt to such changes.

Overall, for the reasons given above the:
Joint Venture Change Agent model is assessed as: Fair to good
for this criteria.

5.8.3.2 Micro-agreements in the Context of Software Project Planning

5.8.3.2.1 Does the relationship model have the capacity to deal with
developments where the schedule is important?
The Captive Model may be useful in achieving delivery of a system within a rigidly defined
timeframe however it does have some limitations which need to be considered.

On the one hand, purely contractual relationship models such as the conventional model
tend to focus the parties on contractually enforceable obligations focused on the products
to be delivered. Those obligations will usually include explicitly stated delivery timeframes
and as such the parties attention will be clearly focused on meeting them. The advantage
therefore of a purely contractual model is that there is usually a degree of certainty in the
obligations of each party, including time for performance type obligations.

With a joint venture type arrangement however the situation potentially may not be so
clear for the parties, particularly if the joint venture objectives do not explicitly state
schedule information. Naturally a joint venture arrangement can impose contractual
obligations on the parties (including obligations relating to schedule) however, as
discussed in earlier Sections, the main focus in a joint venture arrangement is on the
relationship aspects and achieving the joint venture objectives. If those objectives do not
explicitly include schedule related information then there is some risk that the efforts of the
members of the joint venture are directed to achieving other goals.

If however the objectives of the joint venture include (for example) meeting defined
schedule, then it is likely that the parties will be bound to achieve that schedule not only
contractually (through the joint venture agreement itself) but also through fiduciary
obligations as well. In that sense a joint venture arrangement can place a heavier onus on
the parties to achieve schedule than through more conventional contractual relationships.

The key point is that provided the obligations in relation to schedule are clear and
consistent with the joint venture objectives then the Joint Venture Change Agent
relationship model is likely to give very good support for achieving schedule related
objectives. If however the exact schedule obligations are not so clear, or they are not
consistent with the joint venture objectives then there is likely to be some risk in achieving
scheduled milestones.

Overall, when using the Joint Venture Change Agent Model it is important therefore to
ensure that any schedule related requirements are explicitly stated as joint venture
objectives in the joint venture agreement, or at least are given heavy emphasis through
separate contracts where possible. Regardless of the mechanism used, it is important to
ensure that schedule obligations are clear and unambiguous, and that ideally some legal
remedies are available in the case of default. The risk of using the Joint Venture Change
Agent Model is that no contractually enforceable obligations will be placed on particular
parties to meet defined schedule unless the joint venture agreement itself includes them
directly or indirectly.

Despite the above, the nature of a joint venture is such that decisions can be made by the
joint venture itself which influence the delivery schedule.

Overall, for the reasons given above the:
Joint Venture Change Agent model is assessed as: Fair to Good for this criteria.

5.8.3.3 Micro-agreements in the Context of Software Project Tracking and Oversight

5.8.3.3.1 Are the overheads low for this relationship model?
This is a relatively sophisticated model in that a new joint venture is set up to provide products and services to the customer and to third party customers as well. From that perspective, the overheads of setting up a joint venture and managing its day to day operations as well as meeting regulatory requirements may be relatively higher than some of the more conventional relationship models previously discussed.

A joint venture itself is not a separate legal entity and accordingly the infrastructure of each member of the joint venture needs to be maintained independently. In the Captive Model (previously discussed) the focus was on a new corporation formed by interested parties and because it is a single legal entity (that is, the new corporation), only one set of infrastructure needs to be maintained for that entity. A joint venture arrangement on the other hand does not have a single legal entity as the focus but rather each member of the joint venture needs to provide management and other infrastructure as appropriate. Some duplication of this infrastructure may result leading to potential inefficiencies and additional costs associated with this relationship model.

Overall, for the reasons given above the:

Joint Venture Change Agent model is assessed as: Poor to Fair (i.e. high overheads) for this criteria.

5.8.3.3.2 Does this relationship model give the customer good visibility of progress?
Because the customer in this model is a member of the joint venture set up to perform the work then it theoretically has very good visibility of progress. In addition, it also has a good ability to influence decisions about the products and services to be delivered. Due to the fact that the customer is a member of the joint venture, it indeed has legal obligations for ensuring the good management of the work with the aim of meeting the overall objectives of the joint venture.

Overall, for the reasons given above the:

Joint Venture Change Agent model is assessed as: Very Good for this criteria.

5.8.3.3.3 Does the relationship model give management good visibility of progress?
Under the Joint Venture Change Agent model, management responsibility for delivery of products and services is likely to fall on either the members of the joint venture collectively, or as otherwise defined in the joint venture agreement. In that sense, the Joint Venture Change Agent model does give good visibility to management through the fact that the management responsibility is contained within the members of the joint venture structure itself. Although there may be some legal boundaries separating management and developers (if in fact these functions are performed by different organisations), a joint venture arrangement in some ways reduces the rigidity of these legal boundaries by allowing the flow of management information to be performed as required to achieve joint venture objectives. Having such an arrangement is likely to lessen the impact of pure contractual, intellectual property or other such legal issues.

Paul P. Parnell
Provided that there is effective management within the joint venture structure then solid visibility of progress should be available to management.

Overall, for the reasons given above the:

Joint Venture Change Agent model is assessed as: Very Good for this criteria.

5.8.3.4 Micro-agreements in the Context of Sub-Contractor Management

5.8.3.4.1 Does the relationship model allow the management of risks effectively?
This relationship model is designed to bring the suppliers of the system, the system integrator, and the customer into a co-operative team environment. The fact that legal boundaries between the parties have been to some degree eliminated (by having customer and supplier representatives in a single joint venture arrangement) allows for an easier flow of information needed to assess risk.

In contrast the conventional and service model for example, the customer is likely to only obtain data as specified in the particular contracts and sub-contracts. For contractual and intellectual property reasons however some restrictions on accessing information needed to fully assess project risks may not be available. A joint venture approach on the other hand may allow whatever data is necessary to assess risks to be made available to the relevant parties within the joint venture structure for the purpose of assessing risks.

Overall, for the reasons given above the:

Joint Venture Change Agent model is assessed as: Good for this criteria.

5.8.3.5 Micro-agreements in the Context of Software Quality Assurance

5.8.3.5.1 Does the relationship model have the capacity to deal with developments where system reliability is important?
System reliability is one quality assurance issue which may be important to the customer. The Joint Venture Change Agent Model may provide some support for producing a quality product if that aim is consistent with the general joint venture objectives.

Quality processes are usually developed over time by organisations depending upon specific needs. Some organisations may "tune" their processes to achieve fast turnaround or low cost items at the expense of other quality attributes such as reliability. The fact that in a Joint Venture Change Agent Model both the customer and the supplier are represented within the same joint venture structure allows the "cultures" of each individual organisation to be merged over time, and for the interests of all parties to be considered. Despite this, if a joint venture is set up for the achievement of a single specific goal (such as the integration and delivery of a new information technology system) rather than as a long term continuous relationship then this longer term quality improvement mechanism may not be available.

For this model there is certainly scope for the development of "quality" products provided that the level of quality required is consistent with the joint venture objectives.

Overall, for the reasons given above the:
Joint Venture Change Agent model is assessed as: Fair to Good for this criteria.

5.8.3.5.2 Does this relationship model require sophisticated developers and management?

The Joint Venture Change Agent Model aims to give the responsibility of development and management decisions to a separate joint venture formed for that purpose. To that extent the customer is relieved of having to use sophisticated techniques to either manage or develop a system, but rather can (in theory) give broad direction to the systems integrator who, as a member of the joint venture, has obligations to ensure the objectives of the joint venture are met. The direction given could, for example, take the form of advice on the customer's business needs both present and future. From that perspective the customer can off-load as much 'sophistication' and responsibility to the other members of the joint venture as is appropriate. In this structure therefore particular expertise or specialised resources can be made available by individual joint venture members in a manner which gives the best chance of achieving the joint venture objectives.

Thus with the Joint Venture Change Agent model the customer does not particularly need sophisticated developers and managers provided that the necessary expertise is available within the other members of the joint venture.

Overall, for the reasons given above the:

Joint Venture Change Agent model is assessed as: Good for this criteria.

5.8.3.6 Micro-agreements in the Context of Software Configuration Management

5.8.3.6.1 Does the relationship model have the capacity to enforce good software configuration management practices?

It has been previously indicated that the configuration management process is an important one but conceptually reasonably straight forward. It is essentially a set of processes designed to allow changes to configured items (such as source code) in a controlled manner. The main issue then is ensuring that the developers and system integrator follow a defined process and strictly adhere to it to the extent necessary to ensure changes to parts of the system are tracked effectively. This can usually be enforced contractually by requiring the developers to meet specified standards such as some of those identified in Appendix A. Software Standards

Overall, the Joint Venture Change Agent model is considered adequate for enforcing good configuration management practices provided that processes are in place and preferably required as a contractual obligation.

Overall, for the reasons given above the:

Joint Venture Change Agent model is assessed as: Fair to Good for this criteria.

5.8.3.7 Assessment of Legal Risks

What capacity does this relationship model have in minimising legal risks? Specifically, what is the likelihood of claims under contract, estoppel, trade practices or other legal doctrines if this relationship model is used?
In the Joint Venture Change agent model, a joint venture is formed by both the customer and the supplier to provide products and services to the customer, and also to provide products and services to third party customers on a commercial basis.

In some ways this relationship model is has attributes associated with some of the other models. Specifically, the joint venture structure set up between the Customer and Systems integrator aims to foster a degree of co-operation and unity of objectives between the parties in a similar fashion to the captive model previously discussed. In addition, as the joint venture also supplies products and services to third parties, this aspect resembles more closely the conventional model arrangement. The legal risks associated with the Joint Venture Change Agent model are therefore similar in nature to risks previously discussed for the conventional model.

Specifically, legal risks associated with using this model include:

- In the Joint Venture Change Agent model, the risk of actions for breach of contract, misrepresentation and estoppel between the Joint Venture and third party customers may be increased slightly over (for example) the captive model due to the difficulty in accurately predicting the cost and schedule associated with the technical work (but see further discussion points on this under the conventional model assessment – section 5.5.3.7 above;
- in the Joint Venture Change Agent model, the risk of breach of contract between the Joint Venture parties and the customer (who is also a party to the joint venture) is minimised because both have an interest in meeting the objectives of the joint venture.

Overall, for the reasons given above:

The Joint Venture Change Agent Model is assessed as fair-good in minimising legal risks.

5.8.3.8 Summary of Assessment – Joint Venture Change Agent Model

The following table summarises the above assessment.

<table>
<thead>
<tr>
<th>Relationship Model Evaluation Criteria</th>
<th>Effectiveness of Relationship Model in coping with micro-agreements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Micro-agreements in the context of Requirements Management</td>
<td>Does the relationship model have a capacity to deal with inadequately defined requirements?</td>
</tr>
<tr>
<td></td>
<td>Does the relationship model have a capacity to deal with an evolving system architecture?</td>
</tr>
<tr>
<td></td>
<td>Does the relationship model have a capacity to deal with a system which needs to grow?</td>
</tr>
<tr>
<td></td>
<td>Does this relationship model have the capacity to allow changes to be easily introduced into the system?</td>
</tr>
<tr>
<td>Micro-agreements in the context of Software Project Planning</td>
<td>Does the relationship model have the capacity to deal with developments where the schedule is important?</td>
</tr>
<tr>
<td>Relationship Model Evaluation Criteria</td>
<td>Effectiveness of Relationship Model in coping with micro-agreements</td>
</tr>
<tr>
<td>--------------------------------------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------</td>
</tr>
<tr>
<td>Micro-agreements in the context of Software Tracking and Oversight</td>
<td>Poor to Fair (See 5.8.3.3.1)</td>
</tr>
<tr>
<td>· Are the overheads low for this relationship model?</td>
<td></td>
</tr>
<tr>
<td>· Does this relationship model give the customer good visibility of progress?</td>
<td>Very Good (See 5.8.3.3.2)</td>
</tr>
<tr>
<td>· Does the relationship model give management good visibility of progress?</td>
<td>Very Good (See 5.8.3.3.3)</td>
</tr>
<tr>
<td>Micro-agreements in the context of Sub-contractor Management</td>
<td>Good (See 5.8.3.4.1)</td>
</tr>
<tr>
<td>· Does the relationship model allow the management of risks effectively?</td>
<td></td>
</tr>
<tr>
<td>Micro-agreements in the context of Software Quality Assurance</td>
<td>Fair to Good (See 5.8.3.5.1)</td>
</tr>
<tr>
<td>· Does the relationship model have the capacity to deal with developments where system reliability is important?</td>
<td></td>
</tr>
<tr>
<td>· Does this relationship model require sophisticated developers and management?</td>
<td>Good (See 5.8.3.5.2)</td>
</tr>
<tr>
<td>Micro-agreements in the context of Software Configuration Management</td>
<td>Fair to Good (See 5.8.3.6.1)</td>
</tr>
<tr>
<td>· Does the relationship model have the capacity to enforce good software configuration management practices?</td>
<td></td>
</tr>
<tr>
<td>Legal Risks</td>
<td>Fair to Good (See 5.8.3.7)</td>
</tr>
<tr>
<td>· What capacity does this relationship model have in minimising legal risks?</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 38 - Evaluation of Joint Venture Change Agent Model Attributes**

**5.8.3.9 Looking at trends – Micro-agreements and the Joint Venture Change Agent Relationship Model**

Overall, when a micro-agreement occurs within the context of the software engineering fundamental "best practice" areas discussed above, then the following findings are made:

**Finding 25:** The Joint Venture Change Agent relationship model is likely to allow micro-agreements to occur, and in fact encourage them because the joint venture structure allows activities to occur which are consistent with the overall objectives of the joint venture.

Paul P. Parnell
Finding 26: The Joint Venture Change Agent relationship model is likely to provide a good degree of flexibility to enable changing circumstances to be managed effectively, through appropriate changes to the software project being dictated by the joint venture itself, of which interested parties are represented. In addition, appropriate software engineering processes can also be put in place.

Finding 27: The Joint Venture Change Agent relationship model is likely to allow the parties a large amount of discretion in whether it accepts and implements "good" micro-agreements (that is, those which are consistent with the overall business objectives of the joint venture).

Finding 28: The Joint Venture Change Agent relationship model is likely to allow interested parties the discretion to reject and not implement "bad" micro-agreements (that is, those which are not consistent with the overall business objectives of the parties).

Finding 29: The Joint Venture Change Agent relationship model is likely to result in an information technology system which is more likely to satisfy customer needs in terms of cost, quality and schedule, provided that firstly, the additional overheads of setting up and managing a separate joint venture to perform the work is justified. Large or complex software based projects may provide this justification. Secondly, a solid software engineering process is in place for managing the complex issues which are likely to arise in larger projects.

Finding 30: In the Joint Venture Change Agent model, the risk of actions for breach of contract, misrepresentation and estoppel between the Joint Venture and third party customers may be increased slightly over (for example) the captive model due to the difficulty in accurately predicting the cost and schedule associated with the technical work.

Finding 31: In the Joint Venture Change Agent model, the risk of breach of contract between the Joint Venture parties and the customer (who is also a party to the joint venture) is minimised because both have an interest in meeting the objectives of the joint venture.
Finding 32: The Joint Venture Change Agent relationship model may be a good choice for information technology developments where the customer recognises that significant technical risk is involved and where business needs and system requirements are likely to change significantly over the life of a project. On the other hand, smaller well defined projects may not require the complexity and cost of setting up and running a joint venture to perform the work.

Despite the above points, it is considered that the best chance of achieving success with this model will occur if appropriate software engineering processes are put in place.

5.8.3.10 Conclusion

The Joint Venture Change Agent model is one where the customer and suppliers form a joint venture to provide products and services to the customer and third parties as well. There are a number of advantages and disadvantages to this approach however in summary, based on information distilled from the above assessment of the model, in order to successfully implement this model:

- the objectives of the joint venture must be consistent with the business needs of both the customer and suppliers
- interested parties need to be represented as members of the joint venture so that specific interests are addressed adequately
- a systems engineering process should be set up within the project to enable complex project issues to be managed
- risks are taken on by joint venture members to a large degree rather than the individual suppliers of components (if those suppliers are part of the captive corporation)
- the model does allow for changes to be made to the system throughout the life of the development and integration

Of fundamental importance to the successful use of this model is the necessity for the joint venture objectives to be identified correctly. Once identified this model can provide the flexibility to accommodate change and manage risk, provided that the change is consistent with the objectives of the joint venture.

Overall, this model has the advantage of giving a large amount of flexibility to the parties to accommodate changing circumstances Micro-agreements which occur within the context of this model can be implemented or rejected easily depending upon the overall. On the other hand however, the freedom to allow changes to the system easily, or at least more easily than in a conventional (or purely contractual environment) may lead to "requirements creep". Requirements creep occurs when changes to the requirements change at a rate faster than that which the developers can keep up with. A solid systems engineering process can help to manage this and other risks.

5.9 THE TENDENCY OF RELATIONSHIP MODELS TO GENERATE MICRO-AGREEMENTS

Are particular relationship models more likely to generate micro-agreements than others? The following table summarises this issue and provides brief reasons that are based on the preceding discussion.
<table>
<thead>
<tr>
<th>Relationship Model</th>
<th>Likelihood of generating micro-agreements?</th>
<th>Reason</th>
</tr>
</thead>
</table>
| Conventional model                 | High                                     | • As the conventional model imposes contractual boundaries between the parties to a contract. Changing requirements are generally accommodated by formal contract changes that may be cumbersome and unwieldy in a dynamic environment.  
• As an example, individuals may prefer to establish micro-agreements between themselves to bypass the formal contract change mechanisms and allow changes to be made more easily and quickly, particularly when the rate of requirements change is high. |
| Service model                       | High                                     | • This is the same as for the conventional model except that an extra “contract layer” may be introduced between the parties - the buyer contracts a prime or systems integrator who subcontracts other parties. This arrangement may still provide a fairly ‘rigid’ approach to coping with change (like the conventional model), and thus inadvertently encourage micro-agreements. |
| Captive model                      | Low                                      | • The captive model eliminates most contractual boundaries between the parties (in the sense that the customer and system integrator form a separate corporation to supply goods and services to the customer). As the new corporation has the customer as a major decision maker, this arrangement is similar to (but not exactly like) an in-house development.  
• In this model, the subject matter of micro-agreements will still be discussed and agreed between the parties although it is less likely that the micro-agreement will be made between parties in different organisations.  
• As a result, decisions made within the single legal entity (i.e. the captive corporation) will not usually result in legal rights and obligations arising between the parties to the micro-agreement. |
| Joint venture change agent model    | Medium                                   | • This is similar to the captive model except that third-parties are involved. This creates additional contractual boundaries for legal rights and obligations to be created between the parties to a micro-agreement. |

Interestingly, the likelihood of micro-agreements occurring appears to be proportional to the number of legal entities involved. If those entities are combined, such as in the captive model, fewer opportunities will arise for micro-agreements to occur. The analogy is that if software is developed entirely in-house within a corporation, then that corporation cannot sue itself for non-performance.

In summary,
Finding 33: The conventional model has a high tendency to generate micro-agreements.

Finding 34: The service model has a high tendency to generate micro-agreements.

Finding 35: The captive model has a low tendency to generate micro-agreements.

Finding 36: The joint venture change agent model has a medium tendency to generate micro-agreements.

5.10 COMPARING THE RELATIONSHIP MODELS

5.10.1 Introduction
The previous Sections discussed individually the effect of particular legal relationships in the context of micro-agreements. How then do the different relationship models compare with one another? In order to enable easier visualisation of the data provided above, below is provided firstly, a table which lists side-by-side the evaluation criteria for each model; secondly, a table listing the totality of individual "findings" for all relationship models; and thirdly, a discussion aimed at consolidated the data.
### 5.10.2 Side-by-side Comparison of Evaluation Criteria

The table below lists side-by-side the results of evaluation criteria for all relationship models examined.

<table>
<thead>
<tr>
<th>Micro-agreements in the context of Requirements Management</th>
<th>Relationship Model Evaluation Criteria</th>
<th>Conventional Model (See 5.5)</th>
<th>Service Model (See 5.6)</th>
<th>Captive Model (See 5.7)</th>
<th>Joint Venture Change Agent Model (See 5.8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Does the relationship model have a capacity to deal with inadequately defined requirements?</td>
<td>Poor (see 5.5.3.1.1)</td>
<td>Poor to Fair (see 5.6.3.1.1)</td>
<td>Good (see 5.7.3.1.1)</td>
<td>Very Good (see 5.8.3.1.1)</td>
<td></td>
</tr>
<tr>
<td>Does the relationship model have a capacity to deal with an evolving system architecture?</td>
<td>Poor to Fair (see 5.5.3.1.2)</td>
<td>Good (see 5.6.3.1.2)</td>
<td>Good (see 5.7.3.1.2)</td>
<td>Good (see 5.8.3.1.2)</td>
<td></td>
</tr>
<tr>
<td>Does the relationship model have a capacity to deal with a system which needs to grow?</td>
<td>Poor (see 0)</td>
<td>Poor (see 5.6.3.3.3)</td>
<td>Fair to Good (see 5.7.3.1.3)</td>
<td>Fair to Good (see 5.8.3.1.3)</td>
<td></td>
</tr>
<tr>
<td>Does this relationship model have the capacity to allow changes to be easily introduced into the system?</td>
<td>Poor to Fair (see 0)</td>
<td>Poor to Fair (see 5.6.3.1.4)</td>
<td>Not assessed</td>
<td>Not assessed</td>
<td></td>
</tr>
<tr>
<td>Micro-agreements in the context of Software Project Planning</td>
<td>Does the relationship model have the capacity to deal with developments where the schedule is important?</td>
<td>Poor to Fair (see 5.5.3.2.1)</td>
<td>Poor to Fair (see 5.6.3.2.1)</td>
<td>Fair to Good (see 5.7.3.2.1)</td>
<td>Fair to Good (see 5.8.3.2.1)</td>
</tr>
<tr>
<td>Micro-agreements in the context of Software Tracking and Oversight</td>
<td>Are the overheads low for this relationship model?</td>
<td>Very Good (see 5.5.3.3.1)</td>
<td>Fair to Good (see 5.6.3.3.1)</td>
<td>Poor (relatively high overheads) (see 5.7.3.3.1)</td>
<td>Poor to Fair (see 5.8.3.3.1)</td>
</tr>
<tr>
<td>Does this relationship model give the customer good visibility of progress?</td>
<td>Poor to Fair (see 5.5.3.3.2)</td>
<td>Poor (see 5.6.3.3.2)</td>
<td>Very Good (see 5.7.3.3.2)</td>
<td>Very Good (see 5.8.3.3.2)</td>
<td></td>
</tr>
<tr>
<td>Does the relationship model give management good visibility of progress?</td>
<td>Poor to Fair (see 5.5.3.3.3)</td>
<td>Fair (see 5.6.3.3.3)</td>
<td>Very Good (see 5.7.3.3.3)</td>
<td>Very Good (see 5.8.3.3.3)</td>
<td></td>
</tr>
<tr>
<td>Micro-agreements in the context of Sub-contractor Management</td>
<td>Does the relationship model allow the management of risks effectively?</td>
<td>Poor (see 5.5.3.4.1)</td>
<td>Good (see 5.6.3.4.1)</td>
<td>Good (see 5.7.3.4.1)</td>
<td>Good (see 5.8.3.4.1)</td>
</tr>
<tr>
<td>Relationship Model Evaluation Criteria</td>
<td>Conventional Model (See 5.5)</td>
<td>Service Model (See 5.6)</td>
<td>Captive Model (See 5.7)</td>
<td>Joint Venture Change Agent Model (See 5.8)</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------------------------------------------</td>
<td>-------------------------------</td>
<td>-------------------------</td>
<td>------------------------</td>
<td>------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Micro-agreements in the context of Software Quality Assurance</td>
<td>Does the relationship model have the capacity to deal with developments where system reliability is important?</td>
<td>Fair to good (See 5.5.3.5.1)</td>
<td>Fair (See 5.6.3.5.1)</td>
<td>Fair to Good (See 5.7.3.5.1)</td>
<td>Fair to Good (See 5.8.3.5.1)</td>
</tr>
<tr>
<td>Does this relationship model require sophisticated developers and management?</td>
<td>Poor to Fair - yes, very sophisticated (See 5.5.3.5.2)</td>
<td>Good (See 5.6.3.5.2)</td>
<td>Good (See 5.7.3.5.2)</td>
<td>Good (See 5.8.3.5.2)</td>
<td></td>
</tr>
<tr>
<td>Micro-agreements in the context of Software Configuration Management</td>
<td>Does the relationship model have the capacity to enforce good software configuration management practices?</td>
<td>Good (See 5.6.3.6.1)</td>
<td>Good (See 5.6.3.6.1)</td>
<td>Fair to Good (See 5.7.3.6.1)</td>
<td>Fair to Good (See 5.8.3.6.1)</td>
</tr>
<tr>
<td>Legal Risk Minimisation</td>
<td>What capacity does the relationship model have to minimise legal risks?</td>
<td>Poor (See 5.6.3.7)</td>
<td>Poor (See 5.6.3.7)</td>
<td>Good (See 5.7.3.7)</td>
<td>Fair to Good (See 5.8.3.7)</td>
</tr>
</tbody>
</table>

**Figure 39 - Comparison of Relationship Models**

### 5.10.3 Full List of Relationship Model Findings

The following table lists all findings for the relationship models examined, particularly in the context of micro-agreements.

<table>
<thead>
<tr>
<th>CONVENTIONAL RELATIONSHIP MODEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finding 1:</td>
</tr>
<tr>
<td>Finding 2:</td>
</tr>
</tbody>
</table>

Paul P. Parnell
<p>| Finding 3: | The conventional relationship model is likely to allow the parties to a contract little discretion to accept and implement &quot;good&quot; micro-agreements (that is, those which are consistent with the overall business objectives of the parties), except by either direct contract change amendments, or by appropriate software engineering processes which are contractually supported. |
| Finding 4: | The conventional relationship model is likely to allow the parties to a contract the discretion to reject and not implement &quot;bad&quot; micro-agreements (that is, those which are not consistent with the overall business objectives of the parties), on the basis that they are not contractually binding however this model in itself provides little support for the inclusion of &quot;bad&quot; micro-agreements though the operation of other legal mechanisms apart from contract (such as estoppel, unconscionability, etc. as discussed in Chapters 3 and 4 above). |
| Finding 5: | Particularly for large systems, the conventional relationship model is likely to result in an information technology system which is less likely to satisfy customer needs in terms of cost, quality and schedule, unless a solid software engineering process is in place, preferably as a contractual requirement on the parties. |
| SERVICE RELATIONSHIP MODEL | |
| Finding 6: | The legal risk of breach of contract is relatively high for the conventional model (compared with some other models) especially if changing requirements cannot be introduced easily into the contracting process. |
| Finding 7: | The conventional model encourages the formation of contracts for information technology products when industry experience shows that it may be difficult to estimate the cost and schedule associated with acquiring these products. Any estimates of cost, schedule or performance of the system which therefore turn out to be incorrect, may possibly lead to actions in misrepresentation, and estoppel (for example). |
| Finding 8: | The conventional relationship model may be the best choice for simpler information technology developments, or ones where there are few parties involved. Regardless, it is considered that the best chance of achieving success with this model will occur if appropriate software engineering processes are required as contractual obligations. |
| Finding 9: | The service relationship model is likely to allow micro-agreements to occur, or at least not prevent them from occurring. |
| Finding 10: | The service relationship model is likely to provide some flexibility to enable changing circumstances to be managed effectively, specifically by either direct contract change amendments, or by appropriate software engineering processes which are contractually supported |
| Finding 11: | The service relationship model is likely to allow the parties to a contract little discretion to accept and implement &quot;good&quot; micro-agreements (that is, those which are consistent with the overall business objectives of the parties), except by either direct contract change amendments, or by appropriate software engineering processes which are contractually supported |
| Finding 12: | The service relationship model is likely to allow the parties to a contract the discretion to reject and not implement &quot;bad&quot; micro-agreements (that is, those which are not consistent with the overall business objectives of the parties), on the basis that they are not contractually binding however this model in itself provides little support for the inclusion of &quot;bad&quot; micro-agreements though the operation of other legal mechanisms apart from contract (such as estoppel, unconscionability, etc. as discussed in Chapters 3 and 4 above) |
| Finding 13: | The service relationship model is likely to result in an information technology system which is less likely to satisfy customer needs in terms of cost, quality and schedule, unless a solid software engineering process is in place, preferably as a contractual requirement on the parties. |
| Finding 14: | The legal risk of breach of contract is relatively high for the service model (compared with some other models) especially if changing requirements cannot be introduced easily into the contracting process. |
| Finding 15: | The service model encourages the formation of contracts for information technology products when industry experience shows that it may be difficult to estimate the cost and schedule associated with acquiring these products. Any estimates of cost, schedule or performance of the system which therefore turn out to be incorrect, may possibly lead to actions in misrepresentation, and estoppel (for example). |</p>
<table>
<thead>
<tr>
<th>Finding 16:</th>
<th>The service relationship model may be the best choice for information technology projects where the customer recognises that significant technical risk is involved requiring the services of a specialist systems integrator. Regardless, it is considered that the best chance of achieving success with this model will occur if appropriate software engineering processes are required as contractual obligations both in the prime contract between the customer and the systems integrator, and also in subcontracts for system components.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAPTIVE RELATIONSHIP MODEL</td>
<td></td>
</tr>
<tr>
<td>Finding 17:</td>
<td>The captive relationship model is likely to allow micro-agreements to occur, and in fact encourage them because there are no legal or contractual boundaries as such between parties who form part of the captive corporation</td>
</tr>
<tr>
<td>Finding 18:</td>
<td>The captive relationship model is likely to provide a good degree of flexibility to enable changing circumstances to be managed effectively, through appropriate changes to the software project being dictated by the corporation management itself, of which interested parties are represented. In addition, appropriate software engineering processes can also be put in place</td>
</tr>
<tr>
<td>Finding 19:</td>
<td>The captive relationship model is likely to allow interested parties a large amount of discretion in whether it accepts and implements &quot;good&quot; micro-agreements (that is, those which are consistent with the overall business objectives of the captive corporation)</td>
</tr>
<tr>
<td>Finding 20:</td>
<td>The captive relationship model is likely to allow interested parties the discretion to reject and not implement &quot;bad&quot; micro-agreements (that is, those which are not consistent with the overall business objectives of the parties)</td>
</tr>
<tr>
<td>Finding 21:</td>
<td>The captive relationship model is likely to result in an information technology system which is more likely to satisfy customer needs in terms of cost, quality and schedule, provided that firstly, the additional overheads of setting up and managing a separate corporation to perform the work is justified. Large or complex software based projects may provide this justification. Secondly, a solid software engineering process is in place for managing the complex issues which are likely to arise in larger projects.</td>
</tr>
<tr>
<td>Finding 22:</td>
<td>In the captive model, the risk of breach of contract between the supplier and customer is minimised because they become part of the same legal entity. Because the supplier and customer in this model are also directors of the corporation supplying the system, they each have significant managerial influence and control over the activities of the corporation to the extent needed to achieve its objectives.</td>
</tr>
<tr>
<td>Finding 23:</td>
<td>With the captive model, both the supplier and customer are directors of the captive corporation and accordingly have obligations as directors. One potential legal risk is that if either party acts in its own self-interest rather than in the interest of the captive corporation then directors duties (including possibly fiduciary obligations) may be breached.</td>
</tr>
<tr>
<td>Finding 24:</td>
<td>The captive relationship model may be a good choice for information technology developments where the customer recognises that significant technical risk is involved and where business needs and hence system requirements are either not fully known, or are likely to change significantly over the life of a project. On the other hand, smaller well defined projects may not require the complexity and cost of setting up and running a separate corporation to perform the work. Despite the above points, it is considered that the best chance of achieving success with this model will occur if appropriate software engineering processes are put in place.</td>
</tr>
</tbody>
</table>

**JOINT VENTURE CHANGE AGENT MODEL**

| Finding 25: | The Joint Venture Change Agent relationship model is likely to allow micro-agreements to occur, and in fact encourage them because the joint venture structure allows activities to occur which are consistent with the overall objectives of the joint venture. |
| Finding 26: | The Joint Venture Change Agent relationship model is likely to provide a good degree of flexibility to enable changing circumstances to be managed effectively, through appropriate changes to the software project being dictated by the joint venture itself, of which interested parties are represented. In addition, appropriate software engineering processes can also be put in place. |
| Finding 27: | The Joint Venture Change Agent relationship model is likely to allow interested parties a large amount of discretion in whether it accepts and implements “good” micro-agreements (that is, those which are consistent with the overall business objectives of the joint venture). |
| Finding 28: | The Joint Venture Change Agent relationship model is likely to allow interested parties the discretion to reject and not implement "bad" micro-agreements (that is, those which are not consistent with the overall business objectives of the parties) |
| Finding 29: | The Joint Venture Change Agent relationship model is likely to result in an information technology system which is more likely to satisfy customer needs in terms of cost, quality and schedule, provided that firstly, the additional overheads of setting up and managing a separate joint venture to perform the work is justified. Large or complex software based projects may provide this justification. Secondly, a solid software engineering process is in place for managing the complex issues which are likely to arise in larger projects |
| Finding 30: | In the Joint Venture Change Agent model, the risk of actions for breach of contract, misrepresentation and estoppel between the Joint Venture and third party customers may be increased slightly over (for example) the captive model due to the difficulty in accurately predicting the cost and schedule associated with the technical work. |
| Finding 31: | In the Joint Venture Change Agent model, the risk of breach of contract between the Joint Venture parties and the customer (who is also a party to the joint venture) is minimised because both have an interest in meeting the objectives of the joint venture. |
| Finding 32: | The Joint Venture Change Agent relationship model may be a good choice for information technology developments where the customer recognises that significant technical risk is involved and where business needs and system requirements are likely to change significantly over the life of a project. On the other hand, smaller well defined projects may not require the complexity and cost of setting up and running a joint venture to perform the work. Despite the above points, it is considered that the best chance of achieving success with this model will occur if appropriate software engineering processes are put in place. |

**TENDENCY OF RELATIONSHIP MODELS TO GENERATE MICRO-AGREEMENTS**

<p>| Finding 33: | The conventional model has a high tendency to generate micro-agreements |
| Finding 34: | The service model has a high tendency to generate micro-agreements |</p>
<table>
<thead>
<tr>
<th>Finding</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>35</td>
<td>The captive model has a low tendency to generate micro-agreements</td>
</tr>
<tr>
<td>36</td>
<td>The joint venture change agent model has a medium tendency to generate micro-agreements</td>
</tr>
</tbody>
</table>

**Figure 40: Full List of Relationship Model Findings**
5.10.4 Consolidated Findings and Conclusion

In consolidating the data listed above the following main points are made:

- The legal relationship between the parties to a large information technology project is likely to have some influence on the outcome of a micro-agreement. The reason for this is primarily the amount of flexibility provided to the parties to either implement or reject the micro-agreement. In large projects new information continually comes to light as the work progresses and micro-agreements are formed as a way of addressing change.
- Flexibility and an ability to cope with change is the key to managing micro-agreements.
- All of the relationship models are likely to involve contracts. The difference between the models is however that for models like the Conventional Model or the Service Model, the focus of the contract is on delivery of the product, whereas other models like the Captive Model and the Joint Venture Change Agent Model focus on the relationship objectives and overall business needs.
- Some legal relationship models provide a large degree of flexibility for coping with change. The Captive Model and the Joint entire Change Agent Model are examples of relationships where the parties have an incentive (and indeed the legal obligation) to achieve common goals, and the freedom to make decisions which are consistent with those goals. The downside of these models is however that that firstly they may be more costly to set up and administer, and be more complex to manage; secondly, a greater degree of flexibility may in fact mean a greater opportunity to make wrong decisions; and thirdly to some extent there may be less certainty of rights and obligations which can be provided through some of the more conventional models.
- The Captive Model or the Joint Venture Change Agent model may thus be a good choice for larger information technology developments where the customer recognises that significant technical risk is involved and where business needs and system requirements are likely to change significantly over the life of a project.
- On the other hand, purely contractual models such as the Conventional Model and the Service Model can provide certainty of rights and obligations to the extent that individual parties are clear on what they have to deliver by when. The downside of this type of model is that they can be overly inflexible to change (but of course this depends upon the wording of the contracts). Any changes identified may need contract amendments which span more than one legal entity and thus may be more difficult to implement than if one of the other relationship models is chosen.
- The key difference in the models is essentially that some (like the Captive Model) give the responsibility for decision making to one legal entity whereas other models (like the Conventional Model) need two or more legal entities to reach a mutually agreeable decision (through, for example, contract).
- Process issues are important regardless of the model used. A solid software engineering process should be established in order to maximise the benefits of any model. Chapter 4 "Micro-Agreements" Part 1: Concepts and Case Studies" discussed this issue in more detail.
- Some models, such as the conventional and service models, are more likely to generate micro-agreements than other models such as the captive model or joint venture change agent model. The reason for this is that the opportunity for micro-agreements to be established is proportional to the number of legal entities involved in the project. For very large information technology developments the captive mode, for example, may eliminate some of the
contractual boundaries that may otherwise exist, and decisions will more likely to be made within legal entities rather then between them.

Overall, as with most endeavours there are certain compromises and value judgements which must be made and of course there can be no absolutely correct answers as to which relationship model is best. Despite this however the above analysis has shown that for a particular elements of focus, some models may be more appropriate than others and the tables provided in the preceding two Sections (Figure 39 - Comparison of Relationship Model, and Figure 40: Full List of Relationship Model Findings) give some guidance on these issues.
6. Conclusion

6.1 SUMMARY OF OBJECTIVES AND OUTCOMES

6.1.1 Summary of Objectives
The objectives of this document were described in the first chapter as being:

- to discuss the general nature of large information technology projects and highlight any common obstacles to success from an engineering and project management perspective (See Chapter 2)
- to discuss the law applicable to information technology contracting (including a number of related legal concepts such as estoppel) which affect legal rights and obligations (See Chapter 3)
- to introduce the new legal concept of "micro-agreements" and identify the benefits and disadvantages of them through a number of case studies specific to the information technology industry (See Chapter 4)
- to examine the legal relationships which may be present in larger information technology developments and to assess the relative impact of micro-agreements on each of these relationship models, (See Chapter 5), and
- to make specific recommendations to allow the benefits of micro-agreements to be maximised and the adverse effects to be minimised.

The points listed above have been addressed in detail stage by stage in the previous chapters. In conclusion this chapter summarises some of the core issues and recommendations.

6.1.2 Summary of Outcomes
In summary, this work:

1. defined the new legal concept of a "micro-agreement"
2. identified potential problems and benefits of micro-agreements through its application to a number of case studies
3. recommended solutions for maximising the benefits and minimising the problems associated with micro-agreements including:
   - linking information technology contracts to software engineering "best practice"
   - using an appropriate legal relationship model, and
   - the development of an industry wide Information Technology Code of Conduct based on the Trade Practices Act, 1974; and
4. examined micro-agreements from a multi-disciplinary viewpoint of engineering, management and law in an attempt to consolidate "best practice" viewpoints from all three areas.

Apart from the above, general discussions were also provided on a number of relevant engineering and legal issues. These were provided in Chapter 2: The Nature of Large Information Technology Projects and Chapter 3: Contract Interpretation and Related Issues.
6.2 PROBLEMS ENCOUNTERED IN INFORMATION TECHNOLOGY SYSTEMS

6.2.1 The Fundamental Issue

We have seen throughout this document that communication between the parties to large information technology projects is paramount to achieving a successful outcome and accordingly, any mechanisms available to enhance this communication are likely to be beneficial. The development of large information technology systems is complex and usually involves many disciplines including engineering, management, and law. Any solutions to problems encountered need to be implemented in a way that ensures the integrity of outcome from the perspective of each of disciplines, while at the same time addressing ‘global’ issues which cross disciplinary boundaries.

In general terms, all of the above perspectives (that is, engineering, managerial, and legal) may be able to provide some solutions to the specific problems arising however despite this, as we have seen in Chapter 2 The Nature of Large Information Technology Projects, major problems can still occur. This document has attempted to examine the problems from the point of view that large software engineering undertakings can constitute a significant risk, both financially and technically, to stakeholders. The fundamental issue examined in this document is, how a collective or more holistic approach which crosses inter-disciplinary boundaries can be used to provide a greater chance of success in large information technology projects.

6.2.2 The Problems of Large Information Technology Projects

In Chapter 2 The Nature of Large Information Technology Projects some of the problems encountered in large information technology projects were discussed. In particular, issues associated with software lifecycles, requirements management, metrics and measurement, and defects were examined. Problems in these and other areas can ultimately crystallise into higher cost, late delivery, or lower quality of products and services.

Surveys have indicated that some of the main causes of the problems appear to be things like:

- poor project objectives
- bad planning
- problems with the introduction of new technology
- inadequate project management methodology
- insufficient senior staff on team, and
- poor supplier performance

Any solutions proposed therefore should address these issues as well as implement other recommendations made through this document. It is generally acknowledged however that software engineering is not an exact science and that in any system, particularly large or complex ones, will encounter problems which result in increased risk. The environment in which large information technology projects is developed invariably involves multiple disciplines and solutions which address the problems need to take this into account.

---

604 Systems engineering process models are beginning to recognise the importance of having an integrated approach to the development of a system by specifically requiring work associated with the integration of the various disciplines affected. For example, see: Software Engineering Institute: A Systems Engineering Capability Maturity Model, Version 1.1, SECM-MM-85-01;CMU/SEI-95-MM-003, November 1995

605 KPMG Peat Marwick survey

Paul P. Parnell

274
6.3 MERGING THE DISCIPLINES

"Technical [people], however various they may be in other respects, have one thing in common: they do not think like lawyers"\(^{606}\)

6.3.1 Perspectives from Three Disciplines

From the engineering or technical perspective, a large system may have in the order of millions of lines of computer software source code which run on one or more machines having clock speeds approaching one billion cycles per second. These machines may be connected via networks spanning the globe and involve the processing of millions of transactions per day.

From a management perspective, the nature of large and complex information technology developments is such that there may be hundreds of people involved in performing the development and integration work. It is also likely that these individuals work in a number of separate organisations, each of which may have its own organisational "culture" focused on specific business objectives. Chapter 2: The Nature of Large Information Technology Projects gave attempted to give an insight into the engineering and management perspectives.

From a legal perspective, the law has developed over time a number of mechanisms for identifying the rights and obligations associated with such interactions. It has also developed remedies that can be applied in appropriate circumstances. A general discussion on relevant legal aspects is provided in Chapter 3: Contract Interpretation and Related Issues.

The issue of whether the different perspectives from the three disciplines can meet to the extent necessary to allow some degree of synergy is an important one. Modern software engineering management practices are beginning to recognise that not only is it important to address the integration of the system components themselves, but also to integrate the disciplines which are relevant. As an example, the Systems Engineering Capability Maturity Model (SE-CMM)\(^{607}\) identifies eighteen key "Process Areas" which are important from a systems engineering perspective. One of these key process areas is the "Integrate Disciplines" process which attempts to formulate a method for the structured integration of issues relevant to a variety of disciplines. In that process area the following "Best Practices" are identified:

- "BP.04.01 Involve the disciplines that are essential to system development in a timely manner.
- BP.04.02 Promote cross-discipline understanding among the developers.
- BP.04.03 Establish methods for interdisciplinary coordination.
- BP.04.04 Establish and use methods for identifying and resolving interdisciplinary issues, and creating integrated solutions.
- BP.04.05 Communicate results of interdisciplinary activities to affected groups.
- BP.04.06 Develop project goals and ensure that all affected groups and individuals are fully aware of them"

---

\(^{606}\) Dr. J. McL. Emmerson QC, *The Understanding of Technical Evidence* 68 ALJR 874


---

Paul P. Parnell
The point is that some attempt is being made within the systems engineering community itself to merge the interdisciplinary boundaries and to formulate strategies suitable for providing a coherent integration.

This document identifies a new concept of "micro-agreement" which to some extent is a tool which can be used to examine cross-discipline issues. How micro-agreements do this is discussed below.

6.4 THE CONCEPT OF "MICRO-AGREEMENTS"

6.4.1 Micro-agreements – The reason for the Concept

Fundamentally, the concept of a "micro-agreement" has been designed to allow the identification of issues which cross the boundaries of three disciplines: law, engineering and (to a lesser degree), management. In this respect the idea behind micro-agreements is to provide a central focus point which can be used to address issues which, despite having individual solutions within each discipline, may be better approached from a holistic or collective perspective. To give an analogy, to some extent micro-agreement are the "hub" which connects the wheels of engineering, management and law.

![Figure 41: Micro-agreements – Linking the Disciplines](image)

Each discipline in itself is complex and dynamic and influences the outcome of the project. At the same time however, the micro-agreement itself influences how the three disciplines are applied. Thus the underlying reason for the concept of micro-agreements is to provide a way of merging the disciplines of engineering, management and law for the purpose of identifying problems and ultimately providing solutions suitable for large information technology projects. Chapter 4: "Micro-Agreements" Part 1: Concepts and Case Studies provided a number of case studies which were designed to "target" specific activities within the software engineering field. These case studies examined the micro-agreements from legal, managerial and engineering perspectives as appropriate and provided specific recommendations.

The use of "best practice" techniques in all three disciplines – engineering, management and the law – provides some benefits individually but are more effective if cognisance is take of them across all three disciplines collectively. As a simple example, an organisation may have effective internal engineering procedures for the conduct of particular activities such as configuration management however if these procedures are elevated to the status of being legal obligations through contract, then they are more likely to still be applied in case of conflicting business needs.
6.4.2 What is a micro-agreement?
The term “micro-agreement” is original to this work. In terms of a definition, Chapter 4: “Micro-Agreements” Part 1: Concepts and Case Studies provided the following:

“micro-agreement” means:

“an agreement or common understanding between two or more parties involved in an information technology or related development or acquisition project, which relates to some aspect of work associated with that project, but is not necessarily expressly or by implication covered by any existing or planned contractual clauses”.

As an simple example of a micro-agreement, in Case Study 3 - Micro-agreements in the Context of Software Project Tracking and Oversight, the micro-agreement consisted of an agreement between the buyer and a developer, (who were not in a direct contractual relationship) to classify a number of requirements individually as “non-essential”.

Although six examples (case studies) are provided in Chapter 4 above, the variety and types of micro-agreement are probably unlimited. Despite this, the six case studies discussed in that chapter were intended to provide a “cross-section” of issues related to fundamental software engineering practices. Micro-agreements which occur within these fundamental software engineering practice areas are likely to thus have the most impact.

6.4.3 What is the effect of Micro-agreements?
We saw in Chapter 4 that the effects of micro-agreements can be various. Some are beneficial while some are not. We saw how some micro-agreements potentially benefited the particular project by simplifying the program interfaces; whereas other micro-agreements caused additional work and hence additional cost and late delivery of the products.

As a specific example, the micro-agreement in Case Study 3 - Micro-agreements in the Context of Software Project Tracking and Oversight, caused problems in the delivered product. The product which was delivered, although useable to some extent, was difficult for the users to operate under normal operational conditions because a number of the features classified as “non-essential”. In that case, additional work needed to be performed under a separate contract at additional cost to the buyer in order to rectify the situation.

6.4.4 Where do micro-agreements get their force?
In the context of this document we have seen that micro-agreements generally result from the interaction which occurs between parties to information technology developments. The micro-agreements themselves usually consist of some form of agreement or common understanding between those parties, but one relevant issue is where micro-agreements get their force. In other words, where do the rights and obligations associated with that micro-agreement come from, and what power or incentive is there to enforce a micro-agreement?

Overall it is likely that the “force” of a micro-agreement comes from a number of different sources including business, engineering, management, legal and other perspectives. A discussion of the legal perspective is provided further down but firstly some brief comments regarding some of the other perspectives is made.
From a business perspective for example, it should be remembered that large information technology developments are usually undertaken in a commercial environment and accordingly other factors are relevant apart from legal issues. As an example, maintaining a good business reputation is likely to be a strong incentive to meeting the obligations associated with micro-agreements, and in practice may provide a greater "incentive" for meeting those obligations than strict legal obligations. One of the reasons for this is that issues which directly affect business reputation may need to be addressed as soon as possible whereas strictly legal issues may not have such a priority because of the longer timeframes associate with legal enforcement.

Other business incentives for performing obligations under micro-agreements include the fundamental business drivers of cost (or profit), schedule, and product quality. For example, if performing a micro-agreement is likely to lead to increased profits, then it is more likely that it will be performed. Business needs will give micro-agreements some of their "force", although in a different context to that of strict legal enforcement, but nevertheless as a powerful incentive for performance of micro-agreement obligations.

Similarly from an engineering perspective, if the performance of a micro-agreement aids the engineering process then that micro-agreement is more likely to be performed. For example, if a micro-agreement related to configuration management practices is performed, it may result in a better ability to control the integration of multiple system components. The "force" of the micro-agreement thus may come partially from the implementation of various engineering standards and process models although these may or may not be legally enforceable depending upon the circumstances of a particular case.

Apart from the perspectives discussed above, micro-agreements can also get their force from the legal system, although there is no single legal concept which covers all possible micro-agreements. Accordingly, the legal force of micro-agreements comes from a variety of legal doctrines (a number of which were discussed in Chapter 3: Contract Interpretation and Related Issues). As an example, some micro-agreements will get there force through contract law if the usual requirements of contract are met (that is: offer and acceptance; valuable consideration (or a contract under seal); intention to create legal relations; legal capacity to contract; genuine consent; and legality of objects).

Apart from pure contract law, many micro-agreements will get legal support from other legal doctrines. The range of applicable doctrines can be wide and examples of this can be seen in the case studies discussed in Chapter 4. A summary of the legal issues examined in those case studies is provided in the following table.

<table>
<thead>
<tr>
<th>Micro-agreement Reference</th>
<th>Legal Issues Examined</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case Study 1 (See Section: 0)</td>
<td>• misrepresentation (common law)</td>
</tr>
<tr>
<td></td>
<td>• misrepresentation (Trade Practices Act, 1974)</td>
</tr>
<tr>
<td></td>
<td>• breach of contract</td>
</tr>
<tr>
<td></td>
<td>• professional negligence</td>
</tr>
<tr>
<td></td>
<td>• tort of inducing breach of contract</td>
</tr>
</tbody>
</table>

606 "Configuration Management" is one systems engineering process area. Other process areas are to: analyse candidate solutions; derive and allocate requirements; evolve system architecture; integrate disciplines; understand customer needs and expectations; verify and validate system; ensure quality; manage configurations; manage risk; monitor and control technical effort; plan technical effort; define organisations systems engineering process; improve organisations systems engineering process; manage product line evolution; manage systems engineering support environment; provide ongoing skills and knowledge; and coordinate with suppliers. Software Engineering Institute: A Systems Engineering Capability Maturity Model, Version 1.1, SECMM-95-01;CMU/SEI-95-MM-003, November 1995.
### Micro-agreement Reference | Legal Issues Examined
--- | ---
Case Study 2 (See Section: 4.6) | • estoppel  
• unconscionable conduct  
• economic duress
Case Study 3 (See Section: 4.7) | • breach of contract  
• estoppel  
• claim in quantum meruit for money had and received  
• constructive trust  
• unconscionable conduct under the Trade Practices Act, 1974
Case Study 4 (See Section: 4.8) | • misleading & deceptive conduct (Trade Practices Act, 1974)  
• breach of contract  
• estoppel
Case Study 5 (See Section: 4.9) | • breach of contract  
• estoppel  
• misleading and deceptive conduct (Trade Practices Act, 1974)
Case Study 6 (See Section: 4.10) | • claim under quantum meruit

**Figure 42: Where do Micro-agreements get their Force?**

It can be seen from the above table that even for the limited number of case studies presented in this document, the range of legal issues is quite wide. The case study facts were chosen in an attempt to provide coverage for some of the most important aspects of software engineering rather than to use legal concepts as a starting point. Despite the wide range of legal issues involved, the term “micro-agreement” provides a convenient classification for the interaction which can occur in information technology projects.

Overall, the legal “force” associated with micro-agreements comes primarily from a range of traditional legal concepts including those discussed above. In some ways however the term “micro-agreements” goes further to defining a special category or class of facts which encapsulates a number of these legal concepts concurrently. In addition it provides a convenient mechanism for linking in other perspectives such as those from business, engineering and management.

### 6.4.5 How can the adverse effects of micro-agreements be minimised and the beneficial effects be maximised?

Although this issue is complex and involves a multitude of “side-issues” as seen in the preceding chapters, the recommendations made previously in this document can be distilled into three primary elements. In summary, these are:

- **Engineering Focus**: link information technology contracts to software engineering “best practice”
- **Legal Focus**: use an appropriate legal relationship model in order to minimise legal risk and maximise the benefits of micro-agreements, and
- **Industry Focus**: the development of a systems integration Code of Conduct based on the Trade Practices Act, 1974 would be beneficial.

Remember the “wheel and hub” diagram shown above in Figure 41: Micro-agreements – Linking the Disciplines? The focusing of each of the above three recommendations has to some extent addressed the merging of disciplines by providing a link between them. Each
of these recommendations has a different focus while at the same time each has the common link of providing a recommendation in respect of micro-agreements. A more detailed discussion of each of these three recommendations (and others) is provided below.

6.5 RECOMMENDATIONS

6.5.1 The Essence of Coping with Micro-agreements

In once sense, the essence of dealing with micro-agreements in information technology projects is to have an ability to cope with change, while at the same time having a degree of control over cost, schedule and quality. This concept was summed up by Bohem in the following terms:

"To avoid the problems associated with ... stakeholder mismatches, gold plating, inflexible point solutions, high-risk downstream capabilities, and uncontrolled developments software projects need a mix of flexibility and discipline." 609

In order to achieve this balance, a number of recommendations were made throughout this document. Reference should be made to the appropriate Sections above for details of these specific recommendations although in summary, the following Sections briefly discuss the three main ones.

6.5.2 Primary Recommendation: Enforce Software Engineering “Best Practice”

Software engineering is a discipline which has evolved significantly over the past few decades. It is also one which is far from being a “pure” science and accordingly its results are not entirely predictable. Despite this, attempts have been made at defining the processes which are fundamental to the success of a development and to provide practical ways of implementing “best practice” 610. Section 4.4.2: What is Software Engineering “Best Practice”? discussed what this means and for the purpose of this document concluded that it could be focused on six fundamental software engineering practices considered important (if not critical) to the success of any large information technology development. In summary these fundamental practise areas are:

- Requirements management
- Software project planning
- Software project tracking and oversight
- Subcontractor management
- Software quality assurance, and
- Software configuration management

Detailed recommendations related to each of these practise areas is provided in the individual case studies addressed in Chapter 4, and which are summarised in Section: 4.13.2. In summary however it is submitted that the key to achieving a more controlled outcome is to link software engineering best practice into the particular relationship model used (for example a simple contract). The benefit of incorporating “process” type activities has been the subject of research and is reasonably well documented in the literature. For example,

610 For example, see: Software Engineering Institute: Capability Maturity Model for Software, Version 1.1, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pa., 1993.
"Software is an enigma: It pervades our lives and our products in increasing proportions, yet we struggle to meet project deadlines and to remain within budget. Many practitioners argue that software is an arcane art and resist attempts to structure and manage the process. Sceptics also suggest that attempts to develop guidelines would be overwhelmed by differences among projects, firms, and cultures. This research suggests that the sceptics are wrong."

Overall, it needs to be understood that "it is virtually impossible to make all the correct requirements and implementation decisions the first time around. Incomplete or inadequate requirements descriptions are a fact of life in large projects and the legal environment must be set up to allow evolutionary improvements in the software artefacts as new information comes to light. Enforcing a solid software engineering process helps to achieve this.

In terms of specific projects measurable benefits seen include:

- "Over five years, Lockheed cut its development costs by 75 percent, its time to market by 40 percent, and its defects by 40 percent."
- "Over six and a half years, Raytheon tripled its productivity and realised an ROI of almost 8 to 1."
- "Bull realised an ROI of 4 to 1 in 4 years."
- Schlumberger realised an ROI of 9 to 1 in 3½ years.
- "NASA's Software Engineering Laboratory cut its average cost per mission by 50 percent and its defect rate by 75 percent over eight years, while dramatically increasing the complexity of software used."
- "Similar results have been reported at Hughes, Loral, Motorola, Xerox, and other companies that have focused on systematically improving their software processes."

In practical terms, how can software engineering best practice be achieved? Naturally it depends upon the practice in question however one way (as discussed in detail in Chapter 4 is to use standards, procedures, and guidelines which are available in the industry and which have some degree of recognition and acceptance. But do standards really improve quality? One commentator answered this question in the following terms:

---


Paul P. Parmell
"I doubt anyone could prove or disprove that software-quality standards ultimately improve the quality of our products. Standards are only one of many factors that influence quality. However, there is ample evidence to suggest that, on balance, standards and their related guides and recommended practices do improve product quality. Although some standards have shortcomings, the software community is better off with imperfect standards than no standards at all".

Some examples of projects where the use of standards has demonstrably improved software product quality include:

- Local area networks. (Ethernet: IEEE 802 LAN standards)

Despite this, care needs to be used in applying standards. This was discussed in detail in Chapter 4 above, particularly in the context of "blindly" calling up standards without thoroughly examining their suitability for the particular project under consideration. In the literature, criticism of standards has primarily centred on issues such as: the fact that software standards often overemphasise process (as opposed to the product); many software standards aren't standards (in the sense that they are not mandated but merely recommended); it is often impossible to measure conformance to software standards; many software standards prescribe, recommend, or mandate the use of technology that has not been validated objectively; and many software standards are simply too big.

Despite these criticisms of standards, the key issue is process. Standards merely provide a convenient way of implementing fundamental processes. They should not be blindly followed but carefully chosen based on the respective needs of the parties involved.

Overall, the case studies in Chapter 4 above described in detail how an effective software engineering process, whether implemented by standards or otherwise, can lead to a better outcome in terms of cost, schedule and quality. Linking software engineering "best practice" (particularly in the six fundamental practice areas listed earlier) is likely on balance to improve quality, reduce cost and decrease delivery timeframes if implemented effectively.

### 6.5.3 Primary Recommendation: Use an Appropriate Legal Relationship model

#### 6.5.3.1 What are Legal Relationship Models?

How can the legal relationship which exists between the parties affect the outcomes of an information technology project, and how do relationship models and micro-agreements interact in practice? Chapter 5: "Micro-Agreements" Part 2: Relationship Models examined this question. Specifically, the following relationship models were examined in that chapter:

---


---

Paul P. Parnell 282
• **Conventional Model** (See 5.5). In this model the customer takes on the role of the systems integrator by calling in, or outsourcing, the expertise and products of various sub-contractors as required.  

• **Service Model** (See 5.6). In this model the customer identifies and uses a specialist systems integrator to control and oversee all development activities. The systems integrator sub-contracts products and services as necessary for the development and integration of the system.

• **Captive Model** (See 5.7). A separate company is formed by both the acquirer and the supplier to provide products and services to the acquirer.

• **Joint Venture Change Agent Model** (See 5.8). A joint venture is formed by both the customer and the supplier to provide products and services to the customer, and also to provide products and services to third party customers.

The examination of each of these models was conducted against a number of criteria designed to assess the likely outcome when a micro-agreement occurs in a number of key software engineering areas. Section 5.4.2 provided a discussion on the method used to evaluate the models. In summary, the evaluation focused on micro-agreements in the context of fundamental software engineering practices (that is, requirements management, software project planning, software tracking and oversight, sub-contractor management, quality assurance, and software configuration management), and evaluation criteria were applied in the specific context. (For a description of the criteria applied see Figure 29: Relationship Model Evaluation Criteria).

6.5.3.2 The effectiveness of Relationship Models in Coping with Micro-agreements

So what did the analysis or relationship models find? In other words, how effective are each of the relationship models in coping with micro-agreements? In one sense, to be “effective” a relationship model should allow the benefits of micro-agreements to be retained but also allow the adverse effects to be rejected. In summary the discussion provided in Chapter 5 indicated that:

• The legal *relationship* between the parties to a large information technology project is likely to have some influence on the outcome of a micro-agreement. The reason for this is primarily the amount of flexibility provided to the parties to either implement or reject the micro-agreement depending upon the circumstances.

• Flexibility and an ability to cope with change is the key to managing micro-agreements.

• All of the relationship models are likely to involve legal contracts to some extent (such as Partnership agreements etc.). One difference between the models is however that in some like the Conventional Model or the Service Model the focus of the contract is on delivery of the *product*, whereas other models like the Captive Model and the Joint Venture Change Agent Model focus on the *relationship* objectives and overall business needs.

• Some legal relationship models provide a large degree of flexibility for coping with change. The Captive Model and the Joint entire Change Agent Model are examples of relationships where the parties have an incentive (and indeed the legal obligation) to work towards achieving common goals, and the freedom to make decisions which are consistent with those goals. The downside of these models is however that that firstly they may be more costly to set up and administer, and be more complex to manage; secondly, a greater degree of flexibility may in fact mean a greater opportunity to make wrong decisions; and

---

thirdly to some extent there may be less certainty of rights and obligations which can be provided through some of the more conventional models.

- The Captive Model or the Joint Venture Change Agent model may thus be a good choice for larger information technology developments where the customer recognises that significant technical risk is involved and where business needs and system requirements are likely to change significantly over the life of a project.

- On the other hand, contractual models which tend to focus more on the product (rather than the relationship) such as the Conventional Model and the Service Model can provide certainty of rights and obligations to the extent that individual parties are clear on what they have to deliver by when. On the negative side however, this type of model can be overly inflexible to change (but of course this depends upon the wording of the contracts). Any changes identified may need contract amendments which span more than one legal entity and thus may be more difficult to implement than if one of the other relationship models is chosen.

- One key difference between some of the models is essentially that some models (like the Captive Model) give the responsibility for a significant part of the decision making to one legal entity (like the Captive organisation) whereas other models (like the Conventional Model) need two or more legal entities to reach a mutually agreeable decision (through contract).

- Process issues are important regardless of the model used. A solid software engineering process should be established in order to maximise the benefits of any model. Chapter 4, "Micro-Agreements Part 1: Concepts and Case Studies" discussed this issue in more detail.

A full list of findings on this topic is provided in Figure 39 - Comparison of Relationship Models, and Figure 40: Full List of Relationship Model Findings.

### 6.5.3.3 Minimising Legal Risks

The choice of legal relationship model is likely to affect the legal risks associated with an information technology development. Managing these risks is naturally important and involves an understanding of the nature of the risks involved. For the four models examined, in summary these risks include the following points. Note that these are not the only legal risks and that other legal risk may be present depending upon the particular facts.

### Conventional Model

- The legal risk of breach of contract is relatively high for suppliers in the conventional model (compared with some other models) especially if changing requirements cannot be introduced easily into the contracting process.

- The conventional model encourages the formation of contracts for information technology products when industry experience shows that it may be difficult to estimate the cost and schedule associated with acquiring these products. Any estimates of cost, schedule or performance of the system which therefore turn out to be incorrect, may possibly lead to actions in misrepresentation, and estoppel (for example).

### Service Model
• The legal risk of breach of contract is relatively high for suppliers and the system integrator in the service model (compared with some other models) especially if changing requirements cannot be introduced easily into the contracting process.
• The service model encourages the formation of contracts for information technology products when industry experience shows that it may be difficult to estimate the cost and schedule associated with acquiring these products. Any estimates of cost, schedule or performance of the system which therefore turn out to be incorrect, may possibly lead to actions in misrepresentation, and estoppel (for example).

Captive Model

• In the captive model the risk of breach of contract between the supplier and customer is minimised because they become part of the same legal entity (that is, the captive corporation). Because the supplier and customer in this model are also directors of the corporation supplying the system, they each have significant managerial influence and control over the activities of the corporation to the extent needed to achieve its objectives.
• With the captive model, both the supplier and customer are directors of the captive corporation and accordingly have obligations as directors. One potential legal risk is that if either party acts in its own self-interest rather than in the interest of the captive corporation then directors duties (including possibly fiduciary obligations) may be breached.

Joint Venture Change Agent Model

• In the Joint Venture Change Agent model, the risk of actions for breach of contract, misrepresentation and estoppel between the Joint Venture and third party customers may be increased slightly over (for example) the captive model due to the difficulty in accurately predicting the cost and schedule associated with the technical work.
• In the Joint Venture Change Agent model, the risk of breach of contract between the Joint Venture parties and the customer (who is also a party to the joint venture) is minimised because both have an interest in meeting the objectives of the joint venture.

In summary therefore, the choice of relationship model can affect both the type of legal risks involved as well as the allocation of these risks to the various parties. The main point to note from this however is that different options exist depending upon the circumstances of each development and the following section provides some pointers to balancing the need to achieve business outcome with the risks involved.

6.5.3.4 Choosing an Effective Relationship Model

In summary, the choice of which relationship model is used may have a significant affect on the success of a project and the risks (including legal and other risks) involved. In summary therefore the following points are relevant to the choice.

• The conventional relationship model may be the best choice for simpler information technology developments, or ones where there are few parties involved, or where the requirements are well known and fully specified. Regardless, it is considered that the best chance of achieving success with this model will occur if appropriate software engineering processes are required as contractual obligations. In larger developments where there is likely to be a significant amount of change to requirements, using this model may increase
the legal risk, particularly in relation to breaches of contract, and possibly misrepresentation and estoppel.

- The service relationship model may be the best choice for information technology projects where the customer recognises that significant technical risk is involved requiring the services of a specialist systems integrator. Regardless, it is considered that the best chance of achieving success with this model will occur if appropriate software engineering processes are required as contractual obligations both in the prime contract between the customer and the systems integrator, and also in subcontracts for system components. In larger developments where there is likely to be a significant amount of change to requirements, using this model may increase the legal risk, particularly in relation to breaches of contract, and possibly misrepresentation and estoppel.

- The captive relationship model may be a good choice for information technology developments where the customer recognises that significant technical risk is involved and where business needs and hence system requirements are either not fully known, or are likely to change significantly over the life of a project. On the other hand, smaller well defined projects may not require the complexity and cost of setting up and running a separate corporation to perform the work. Despite the above points, it is considered that the best chance of achieving success with this model will occur if appropriate software engineering processes are put in place. This model is effective at minimising legal risk compared to some of the other models although it may involve additional cost and management effort to establish.

- The Joint Venture Change Agent relationship model may be a good choice for information technology developments where the customer recognises that significant technical risk is involved and where business needs and system requirements are likely to change significantly over the life of a project. On the other hand, smaller well defined projects may not require the complexity and cost of setting up and running a joint venture to perform the work. This model is effective at minimising legal risk compared to some of the other models although it may involve additional cost and management effort to establish.

Overall, in order to maximise the benefits of micro-agreements and to minimise the any adverse effects it is recommended that an appropriate legal relationship model be used.

6.5.4 Primary Recommendation: Develop an Industry Code of Conduct

The case studies presented in Chapter 4 identified some of the more important issues to be addressed in information technology projects including fundamental software engineering practices. The particular practises focused on include: requirements management, software project planning, software project tracking and oversight, subcontractor management, software quality assurance, and software configuration management.

The issue of disclosure of the level of these activities is important if valid and realistic decisions are to be made in acquiring information technology systems. In order to provide some legal support for such disclosure, it is recommended that a new Code of Conduct be established under the Trade Practices Act, 1974\(^{621}\) (Section 4.11 above discussed this issue in more detail).

The primary aims and desired outcomes of the (proposed) Information Technology

---

\(^{621}\) The Trade Practices Act, 1974 Part IVB provides for the legal recognition of industry codes of conduct. One example of such a code is: Department of Workplace Relations and Small Business: Franchising Code of Conduct, 1 July 1998
Systems Integration and Outsourcing Code of Conduct are:

- to encourage fair dealing between suppliers and buyers of information technology products and services, within the context of existing trade practices laws
- to improve communication and strengthen relationships between suppliers and buyers of information technology products and services
- to provide buyers of with some degree of visibility of the processes used by suppliers so that it can make a more accurate assessments of risk
- to provide a balanced basis for comparing between different suppliers and outsourcers.

An detailed outline for a new code of conduct is provided above in Section 4.11.5.2 Outline of Proposed Code of Conduct above and that Section provides further details of the concept. The main focus of the code is to allow for disclosure of relevant issues in the context of a code that is suitable for the information technology industry. Specifically, that outline includes disclosure of the following software engineering fundamental best practise areas:

- Requirements management practices
- Software project planning practices
- Software project tracking and oversight practices
- Subcontractor management practices
- Software quality assurance practices
- Software configuration management practices

In terms of development and implementation of the code, it is recommended that an administrative body such the Commonwealth Department of Communications Information Technology and the Arts provide the managerial and administrative support for development and implementation of the code. An expert advisory body should be identified (or established) to provide recommendations and analysis of proposals for the code; and a method of trialing and measuring the effects of the code should be established before final implementation.

Once the code has been developed and used on a trial basis, it is recommended that it be given force as an industry code of conduct under the Trade Practices Act, 1974 Part IVB. The code should initially be implemented as a voluntary code to enable sufficient data to be collected on its suitability and appropriateness for achieving the primary aims referred to above.

Any code is likely to result in additional work for the parties involved and accordingly it is important that issues of efficiency be addressed. The "bottom line" is thus whether the overall benefits derived from the code outweigh the costs. It is submitted that the primary way of achieving a reasonable return on investment is to link the code to fundamental software engineering best practise areas where there is a quantifiable benefit. By having industry parties disclosure their use of such best practices, it is likely that the software engineering community as a whole will at least become more aware of the current state-of-the-art practices. The risk is of course that any best practise identified (directly or indirectly) in the code may become outdated. In order to overcome this issue firstly there needs to be a mechanism for regular review and update. Despite this however the example outline code developed in this document covers best practise areas which have
been relatively stable for a number of years and are likely to remain so for the foreseeable future\textsuperscript{622}.

Overall, it is submitted that the development of an “Information Technology Systems Integration and Outsourcing Code of Conduct” will allow parties to make better informed decisions about software acquisition; raise the level of awareness of the fundamental best practice areas within the software industry; and encourage fair dealing between suppliers and buyers of information technology products and services; all within the context of existing trade practices (and related) laws.

### 6.6 CONCLUSION

Micro-agreements play an important role in any large software development because it is only through continuous effective communication between the parties involved that new issues which invariably come to light will be efficiently resolved.

It is considered that the recommendations made in this document, when implemented appropriately, are likely to provide a good balance between providing sufficient flexibility to accommodate change (including micro-agreements), while at the same time providing an appropriate degree of stability and control over cost, schedule and quality.

Regardless of the solutions proposed it must be recognised that neither the law nor software engineering provide all the answers in every case. We can only attempt to develop solutions in those fields which on balance are likely to increase the probability of success. Ultimately the appropriateness of particular solutions will depend on the circumstances of the individual case. In this light we should always remember that:

- “We are obliged to cope with a world full of complexity, unknowns, limitations, mistakes, and general imperfection.
- People are by far the most variable and vital components of software projects.
- Everything has a cost, and what we want always exceeds what we can afford.
- Quality is ultimately situational and subjective.
- To achieve excellence in something as complex as software, we have to solve a lot of difficult problems, make a lot of trade-offs, and resolve contradictory values. Excellence does not come easily or mechanically.”\textsuperscript{623}

\textsuperscript{622} The Software Engineering Institute: Software Capability Maturity Model has remained relatively stable since at least 1987 (see bibliography) and at an industry conference held in Canberra April 1999 it was confirmed that the Software CMM is still one of the most popular software engineering process models: SEA’99 Software Engineering Australia Conference - Improving the Business of Software, 12-14 April 1999.

THE END
7. Appendix A. Software Standards

The following table is an extract from MIL-STD-498 *Software Development and Documentation*. It lists various standards related to aspects of software development. Note that MIL-STD-498 covers aspects of all standards listed below and despite being originally a military standard, it has now been commercialised and is known as ISO/IEC 12207 *Information Technology - Software Life-cycle Processes*, 1995-08-01. Specific standards (and indeed specific clauses with those standards need to be evaluated for appropriateness depending upon the circumstances of each project).

Note that other standards exist apart from those listed in the table below. In addition, standards development and improvement is usually an ongoing activity and as such an investigation as to what standards are currently available should be conducted prior to any significant development.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Standards</th>
</tr>
</thead>
<tbody>
<tr>
<td>Behavioural Design</td>
<td>MIL-STD-1801, User Computer Interface</td>
</tr>
<tr>
<td></td>
<td>MIL-HDBK-761, Human Engineering Guidelines for Management Information Systems</td>
</tr>
<tr>
<td>Computer Security</td>
<td>DOD-5200.28 STD, DoD Trusted Computer System Evaluation Criteria</td>
</tr>
<tr>
<td></td>
<td>ANSI/IEEE Std 1042, Guide to Software Configuration Management</td>
</tr>
<tr>
<td></td>
<td>MIL-STD-973, Configuration Management</td>
</tr>
<tr>
<td></td>
<td>MIL-HDBK-61, Guidelines for Configuration Management</td>
</tr>
<tr>
<td>Continuous Acquisition and Lifecycle</td>
<td>MIL-STD-1840, Automated Interchange of Technical Information</td>
</tr>
<tr>
<td>Support (CALS)</td>
<td>MIL-STD-1556, Government-Industry Data Exchange Program</td>
</tr>
<tr>
<td></td>
<td>MIL-HDBK-59, Continuous Acquisition and Lifecycle Support Program Implementation Guide</td>
</tr>
<tr>
<td></td>
<td>MIL-HDBK-600, Documentation Streamlining</td>
</tr>
<tr>
<td></td>
<td>MIL-D-28000, Digital Representation for Communication of Product Data</td>
</tr>
<tr>
<td></td>
<td>MIL-M-28001, Mark-up Requirements and Generic Style Specification for Electronic Printed Output</td>
</tr>
<tr>
<td></td>
<td>and Exchange of Text</td>
</tr>
<tr>
<td></td>
<td>MIL-M-28002, Requirements for Raster Graphics Representation in Binary Format</td>
</tr>
<tr>
<td></td>
<td>MIL-D-28003, Digital Representation for Communication of Illustration Data.</td>
</tr>
<tr>
<td>Joint Technical and Management Reviews</td>
<td>ANSI/IEEE Std 1016, Recommended Practice for Software Design Descriptions</td>
</tr>
<tr>
<td></td>
<td>MIL-STD-499, Engineering Management</td>
</tr>
<tr>
<td>Programming Languages</td>
<td>ANSI/ISO/IEC 8652; Ada</td>
</tr>
</tbody>
</table>

624 from MIL-STD-498 *Software Development and Documentation*, 5 December 1994 Figure 2 at page 30.
<table>
<thead>
<tr>
<th>Category</th>
<th>Reference</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IEEE/ANSI Std 990, Recommended Practice for Ada as a Program Design Language</td>
<td>DOD-STD-1467, Software Support Environments</td>
</tr>
<tr>
<td>Software Development Environment</td>
<td>IEEE STD 1209, Recommended Practice for the Evaluation and Selection of CASE Tools</td>
<td>MIL-HBK-782, Software Support Environment Acquisition</td>
</tr>
<tr>
<td></td>
<td>MIL-HDBK-498, Guidebook on MIL-HDBK-498</td>
<td>ISO/IEC 9126, Quality Characteristics and Guidelines for their Use</td>
</tr>
<tr>
<td>Software Requirements</td>
<td>ANSI/IEEE STD 830, Recommended Practice for Software Requirements Specifications</td>
<td>IEEE STD 1298/892, System Safety Program Requirements</td>
</tr>
<tr>
<td>Systems Engineering</td>
<td>MIL-STD-499, Engineering Management</td>
<td></td>
</tr>
</tbody>
</table>
### Appendix A: Software Standards

<table>
<thead>
<tr>
<th>Tailoring</th>
<th>MIL-HDBK-248, Guide for Application and Tailoring of Requirements for Defense Material Acquisitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>MIL-STD-1379, Military Training Programs</td>
</tr>
</tbody>
</table>

The full list of IEEE software engineering standards is as follows:

<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>830-1993 IEEE Recommended Practice for Software Requirements Specifications (ANSI)</td>
</tr>
<tr>
<td>990-1987 (R1992) IEEE Recommended Practice for Ada As a Program Design Language (ANSI)</td>
</tr>
<tr>
<td>1008-1987 (R1993) IEEE Standard for Software Unit Testing (ANSI)</td>
</tr>
<tr>
<td>1016-1987 (R1993) IEEE Recommended Practice for Software Design Descriptions (ANSI)</td>
</tr>
<tr>
<td>1042-1987 (R1993) IEEE Guide to Software Configuration Management (ANSI)</td>
</tr>
<tr>
<td>1044-1993 IEEE Standard Classification for Software Anomalies (ANSI)</td>
</tr>
<tr>
<td>1044-1993 IEEE Standard Classification for Software Anomalies together with</td>
</tr>
<tr>
<td>1044.1-1995 IEEE Guide to Classification for Software Anomalies</td>
</tr>
<tr>
<td>1062-1993 IEEE Recommended Practice for Software Acquisition (ANSI)</td>
</tr>
<tr>
<td>1209-1992 IEEE Recommended Practice for the Evaluation and Selection of CASE Tools (ANSI)</td>
</tr>
<tr>
<td>1220-1994 IEEE Trial-Use Standard for Application and Management of the Systems Engineering Process</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Document Number</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1348-1995</td>
<td>IEEE Recommended Practice for the Adoption of Computer-Aided Software Engineering (CASE) Tools (ANSI)</td>
</tr>
</tbody>
</table>
8. Appendix B - Glossary

The following definitions, unless otherwise indicated, are extracted from IEEE Standard 610.12-1990 Glossary of Software Engineering Terminology (ANSI).

Baseline
(1) A specification or products that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development, and that can be changed only through formal change control procedures.
(2) A document or set of documents formally designated and fixed at a specific time during the life cycle of a configuration item.
(3) Any agreement or result designated and fixed at a given time, from which changes require justification and approval.

Baseline Management
In configuration Management, the application or technical and administrative direction to designate the documents and changes to those documents that formally identify and establish baselines at specific times during the life of a configuration item.

CASE
Acronym for Computer Aided Software Engineering

CCB
Acronym for Configuration Control Board.

Compile
To translate a computer program expressed in a high order language into its machine language equivalent.

Compiler
A computer program that translates programs expressed in a high order language into their machine language equivalents.

Computer language
A language designed to enable humans to communicate with computers.

Configuration Control
An element of Configuration Management, consisting of the evaluation, co-ordination, approval or disapproval, and implementation of changes to configuration items after formal establishment of their configuration identification.

Configuration Control Board
A group of people responsible for evaluating and approving or disapproving proposed changes to configuration items, and for ensuring implementation of approved changes.

Configuration Item
An aggregation of hardware, software, or both, that is designated for configuration management and treated as a single entity in the configuration management process.

Configuration Management
A discipline applying technical and administrative direction and surveillance to: identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specified requirements.

Incremental Development
A software development technique in which requirements definition, design, implementation, and testing occur in an overlapping, iterative (rather than sequential) manner, resulting in incremental completion of the overall software product.
Interface Requirement: A requirement that specifies an external item with which a system or system component must interact, or that sets forth constraints on formats, timing, or other factors caused by such an interaction.

Metric: A quantitative measure of the degree to which a system, component, or process possesses a given attribute.

Software Engineering: (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.

Software Life Cycle: The period of time that begins when a software product is conceived and ends when that software is no longer available for use. The software life cycle typically includes a concept phase, requirements phase, design phase, implementation phase, test phase, installation and checkout phase, operation and maintenance phase, and, sometimes, retirement phase. Notes: (1) These phases may overlap or be performed iteratively.

Spiral model: A model of the software development process in which the constituent activities, typically requirements analysis, preliminary and detailed design, coding, integration, and testing, are performed iteratively until the software is complete.

Waterfall Model: A model of the software development process in which the constituent activities, typically a concept phase, requirements phase, design phase, implementation phase, test phase, and installation and checkout phase, are performed in that order, possibly with overlap but with little or no iteration.
9. Appendix C - List of Software Engineering Activities

This Appendix describes briefly the software engineering activities as defined by the Capability Maturity Model\(^{626}\). This model was designed to allow the assessment of an organisation's software engineering capability by identifying specific attributes. The model identifies five levels of capability as follows:

- Level 1 - Initial
- Level 2 - Repeatable
- Level 3 - Defined
- Level 4 - Managed
- Level 5 - Optimised

The following paragraphs describe briefly the capability assessment areas identified by the Capability Maturity Model. The model itself consists of a series of questions intended to assess the effectiveness of an organisation in software development activities.

9.1 ORGANISATION AND RESOURCE MANAGEMENT

9.1.1 Organisational Structure
This set of questions covers issues such as the appointment and responsibilities of the software manager, quality assurance manager, and configuration manager.

9.1.2 Resources, Personnel and Training

This Section address physical resource availability (such as workstations) as well as required training programs for particular roles.

9.1.3 Technology Management

This part looks at how an organisation maintains an awareness of emerging technologies and keeps up to date.

9.2 SOFTWARE ENGINEERING PROCESS AND ITS MANAGEMENT

9.2.1 Documented Standards and Procedures

Questions in this Section address issues related to standardisation. It covers tools and techniques, management review, documentation, software reuse, code maintainability, design reviews, estimating, user interface development and many other issues applicable to a software engineering process.

9.2.2 Process Metrics

This Section covers the measurement of the software process itself including cost, schedule, and quality measures. It lists specific profiles and statistics which should be gathered.

9.2.3 Data Management and Analysis

The storage of collected process metrics is addressed by this Section. The analysis of data is also prescribed in order to maximise visibility into tracking an oversight activities.

9.2.4 Process Control

This Section lists attributes and mechanisms for controlling the software process itself. It lists identifying and resolving issues associated with the process including the review of products and compliance with standards. This is aimed at allowing a continual improvement of processes over time.
| Best Practice | Timebox Development | This technique emphasises the priority of the schedule and allows developers a “timebox” in which to develop the required functionality. At the end of a “timebox” the aim is to have a working product, perhaps with more limited functionality that would be possible without the time constraints. |
| Best Practice | Tools Group | This is a group of people whose function is to identify, evaluate “tools” to be used by software developers. Having this function separate from the main development work can avoid duplication of effort in a multi-project environment. |
| Best Practice | Top-10 Risks List | This is a practice for managing risk. Simply, it consists of regularly undertaking a risk assessment and identifying the top ten risks to the project. Once the risks are identified, explicit risk mitigation planning can be undertaken on these risks. |
| Best Practice | User-Interface Prototyping | The user interface can be one of the most dynamic parts of any system and may change significantly once the users actually start to use it. User-interface prototyping can be used to quickly develop a product which the users can evaluate. |
| Best Practice | Voluntary Overtime | As a management technique, in order to decrease the development schedule of a product, voluntary staff overtime can be a significant factor, depending upon the development. |
11. Bibliography

A

Akindemowo, Olujoke, Information Technology Law in Australia, First Edition, LBC Information Services, 1999

Anthes, Gary: "No More Creeps", Computerworld, May 2, 1994

Atiyah, P.S. "Duress and the Overborne Will Again", (1983) 99 LQR 353


Blair, Colin: "Collins' Platform is Fine but Combat System still needs Tuning", Asia-Pacific Defence Reporter, September-October 1996


Brooks, Federick P. Jr. The Mythical Man-Month, 1975, Addison-Wesley

Butt, P. (editor) *Practical Forms and Precedents NSW*, Butterworths, (looseleaf service)


1988 Chambers Science and Technology Dictionary

Commonwealth Department of Administrative Services: *Restricted Panel of Systems Integrators Request for Tender* issued 2 December 1991

Commonwealth of Australia, Government Information Technology Conditions "GITC" Term Agreement, Version 2, 1 December 1994 (see e.g. URL: http://www.gems.gov.au)


DeMarco, T. Controlling Software Projects, Yourdon Press, Englewood Cliffs, N.J.


Elliott, Paul: "Developments in Equitable Estoppel since Walton's *Surex*", being a paper delivered at the BLEC Workshop: "The Law of Contract Disputes, 1994"

Evans, Michael: *Outline of Equity and Trusts*, Butterworths, 1998


Glib, Tom: *Software Metrics*, Winthrop, 1977


Hall, Peter, M.: *Unconscionable Contracts and Economic Duress*, 1985, CCH Australia


Henty, K. *Systems Integration Contracts*, Proc NSW SCL 9 (1992) 1


Hughes, G.L & Sharp, A: *Computer Contracts – Principles and Precedents*, LBC, 1987 (see also the looseleaf version)

Humphrey, Watts as cited in Sydney Morning Herald article "One Mans Quest for Perfection", 3 September 1995 p3c


Khoury, D. and Yamouni, Y.S. "Understanding Contract Law", Butterworths, 1985


Kersher, Bruce, and Noone, Michael: Remedies, 2nd Ed., The Law Book Company Limited, 1990


Knight, P. and Fitzsimons, J: Negotiating Information Technology Contracts, being papers presented at a seminar held Sydney 14,15 September 1995


Lewison, K. The Interpretation of Contracts, Sweet & Maxwell, London, 1989


Strehlo, Kevin: "Catching up with the Joneses and 'Requirement' Creep", InfoWorld, 29 July 1996


United States Department of Defence: Cost Schedule Control Systems, Instruction 5000.2


Whitmore, Sam: "Readers Shed Development Woes", PC Week, 29 May 1995

Zumbo, Frank: "Unconscionability Within Commercial Transactions", being seminar materials from a seminar presented in Sydney March 1996, Legal and Accounting Management Seminars Pty. Ltd. (LAAMS)
12. **Index**

Agency, 38, 71, 188
Best practice, ii, 82, 85, 86, 87, 88, 90, 115, 122, 124, 127, 130, 137, 151, 159, 166, 173, 177, 181, 220, 232, 244, 258, 1, 4, 7, 8, 9, 10, 16, 1
Breach of Contract, 39, 110, 132, 134, 135, 146, 146, 249, 6, 7
Change Control Board, 20
Code of Conduct, ii, 113, 117, 123, 125, 136, 138, 149, 151, 158, 159, 165, 166, 168, 169, 170, 172, 177, 178, 179, 180, 181, 1, 7, 14, 15, 2
Coding, 12, 17, 24, 118, 122, 2, 1
Coercion, 38, 135
Collateral Contracts, 30, 56, 58
Configuration Management, 70, 89, 142, 161, 162, 163, 164, 165, 167, 168, 172, 173, 200, 203, 217, 220, 230, 232, 242, 244, 256, 258, 264, 4, 8, 11, 14, 15, 1
Consideration, 9, 20, 38, 43, 49, 50, 51, 52, 53, 55, 56, 57, 60, 74, 80, 83, 84, 97, 107, 132, 133, 141, 143, 145, 163, 188, 195, 6, 10
Constructive trust, Trust
Constructive, 134, 135, 194
Contract
Affirmation, 36, 39, 40, 59, 110
Implied terms
Class of Contract, 71
Custom, 103, 104
Standard of Performance, 74, 75
Trade Practices Act, 1974, 76
Warranty, 34, 35, 57, 58, 59, 60, 65, 78, 79, 101
Cost, 1, 3, 9, 16, 17, 18, 20, 21, 23, 24, 27, 31, 33, 34, 37, 86, 87, 89, 91, 92, 93, 94, 96, 100, 105, 114, 118, 120, 121, 122, 127, 129, 130, 135, 137, 139, 140, 141, 142, 150, 153, 155, 156, 158, 168, 174, 176, 181, 182, 184, 196, 198, 199, 205, 208, 209, 212, 213, 214, 216, 221, 222, 225, 227, 229, 233, 234, 236, 237, 241, 245, 246, 249, 250, 252, 255, 259, 260, 265, 266, 267, 268, 269, 2, 5, 8, 9, 10, 14, 16, 2, 1, 2
Deceptive Conduct, 47, 61, 144, 145, 146, 148, 150, 157, 160, 176, 199, 7
Defects, 3, 20, 24, 27, 156, 157, 9
Definitions, 7, 185, 189, 191, 196, 197
Duress, 30, 37, 38, 39, 40, 99, 110, 132, 135
Economic, 37, 38, 39, 99, 110, 132, 135, 6
Economic, Economic Duress, 37, 38, 39, 99, 110, 132, 135, 6
to Goods, 37, 38, 39, 110
to the Person, 37
Duty to Co-operate, 75, 112, 113, 116, 178
Duty to Co-operate, Co-operation, 75, 112, 113, 116, 178
Basic Principles, 46, 106
by Acquiescence, 46, 47
Common Law Estoppel, 46, 47, 48, 53, 54, 109
Equitable Estoppel, 46, 48, 49, 50, 51, 52, 53, 54, 107, 109
Promissory Estoppel, 46, 47, 48, 49, 51, 52
Proprietary Estoppel, 46, 47
Evaluation Criteria, 141, 201, 202, 204, 223, 262, 263, 11
Evidence
Extrinsic, 63, 103
Extrinsic, Extrinsic evidence, 63, 103
Parol, 56, 57, 61, 62, 63, 64, 103
Parol, Parol Evidence, 56, 57, 61, 62, 63, 64, 103
Extrinsic evidence, 63, 103
Fiduciary, 30, 31, 134, 145, 188, 190, 191, 193, 194, 195, 249, 250, 251, 253
Fiduciary obligations, 30, 31, 134, 145, 188, 190, 191, 193, 194, 195, 249, 250, 251, 253
Findings, 113, 123, 136, 149, 158, 165, 174, 177, 220, 233, 245, 268, 284, 270, 271, 272, 12
Function Points, 20, 24, 25, 26, 27

Paul P. Parnell  
Index 7 -
Information Technology Law

United States v. European Community, 78, 80
United States v. May, 204
United States v. Mendez, 204
United States v. Nadeau, 204
United States v. Paladino, 204
United States v. Pappas, 204
United States v. Shannon, 204
United States v. St. John, 204
United States v. St. Louis, 204
United States v. Turoff, 204
United States v. Vassallo, 204
United States v. Weitzenkorn, 204
Unrelated Language, 12, 13, 14
Unrelated Sample, 12, 13, 14
Unrelated Test, 12, 13, 14
Unrelated Text, 12, 13, 14
Unrelated Unit, 12, 13, 14
Unrelated Work, 12, 13, 14

Paul P. Parnell
Unconscionable Contracts, 30
Undue influence, 30, 40, 41, 43, 45, 111, 112
Presumption, 41
Unjust Enrichment, 39, 79, 80, 123, 133, 134, 135, 164, 7
Waterfall Lifecycle Model, 10, 11, 118, 121, 122