Software Design Measures for
Distributed Enterprise Information Systems

by
PABLO HERNÁN ROSSI

A thesis submitted for the degree of Doctor of Philosophy

School of Computer Science and Information Technology,
Faculty of Applied Science,
RMIT University, Melbourne, Australia.

May 2004
Declaration

I certify that

☐ except where due acknowledgement has been made, this work is that of the author alone;

☐ this work has not been submitted previously, in whole or in part, to qualify for any other academic award;

☐ the content of this thesis is the result of work which has been carried out since the official commencement date of the approved research program;

☐ any editorial assistance carried out by a third party is acknowledged.

Pablo Rossi
May 2004
Acknowledgements

First and foremost, I owe a great deal to Associate Professor George Fernandez, who has been my senior supervisor during the whole candidature. He helped to sharpen and focus my research over the months, while letting me define my own direction. I will always cherish my collaboration with him. I am also grateful to Associate Professor James McGovern for his continuous support.

My family has been largely helpful in my education. My parents always encouraged me to excel and taught me to value hard work. They routinely sacrificed their needs to meet mine. Without them, I would have never been here.

Of course this would have been very hard without the company of my wife Veronica. She willingly gave up her career, and put up with the schedules of a Ph.D. student without hesitation. My daughter Camila, who entered our life in the middle of the project, has made the ending of this dissertation very special. My only regret is that some of my family and friends are not here to share this with me.

Finally, I would like to thank to the National Technological University (Santa Fe, Argentina) and to RMIT University (Melbourne, Australia) for providing financial support to undertake this project.
## Contents

### Summary 1

### 1 Introduction 3

1.1 Motivation ........................................ 3

1.1.1 Software ...................................... 3

1.1.2 Engineering software .......................... 5

1.1.3 Measurement in software engineering .......... 7

1.2 Overview of this dissertation ....................... 8

1.2.1 Objectives of this dissertation .................. 9

1.2.2 Outline of the rest of the dissertation ........... 10

### 2 Background and Related Work 11

2.1 Introduction ...................................... 11

2.2 Distributed Information Systems .................... 11

2.2.1 Middleware ................................... 13

2.2.2 Enterprise Information Systems ................. 16

2.2.3 Software Engineering with Distributed Technologies .... 17

2.3 Software Measurement ............................ 19

2.3.1 Measurement Theory .......................... 21

2.3.2 Software Product Quality ...................... 22

2.3.3 Software Design Measures ..................... 23

2.4 State of the art of software measurement for DIS .... 30

2.4.1 Shatz [1988] .................................. 30

2.4.2 Cheng [1993] .................................. 32
CONTENTS

2.4.3 Tsaur and Horng [1998] .................................. 33
2.4.4 Morasca [1999] ........................................ 35
2.4.5 Discussion ............................................. 36
2.5 Summary .................................................. 37

3 Methodology ................................................ 39
3.1 Introduction .............................................. 39
3.2 Description of the Research Method ................. 40
  3.2.1 Informal Model .................................... 40
  3.2.2 Formal Model ...................................... 40
  3.2.3 Definition of Measures and Theoretical Validation 44
  3.2.4 Empirical Evaluation ............................. 47
  3.2.5 Practical Applicability Analysis ............... 51
3.3 Summary .................................................. 52

4 Measurement Models ....................................... 53
4.1 Introduction ............................................. 53
4.2 Informal Model ........................................ 53
  4.2.1 Structure .......................................... 54
  4.2.2 Behaviour ........................................ 57
  4.2.3 Summary .......................................... 71
4.3 Formal Model ........................................... 72
  4.3.1 Measurement Goals ............................... 72
  4.3.2 Experimental Hypotheses ....................... 74
  4.3.3 Entity Abstractions .............................. 74
4.4 Summary ................................................ 79

5 Measure Definition and Theoretical Validation ........ 83
5.1 Introduction ............................................ 83
5.2 Distance Framework .................................... 83
  5.2.1 A metric space for sets ......................... 84
  5.2.2 An example ..................................... 85
5.3 Structural Coupling .................................... 86
5.3.1 Cluster Perspective ................................................. 87
5.3.2 Component perspective ........................................... 90
5.3.3 Dependency Perspective ........................................... 93
5.4 Behavioural Coupling ................................................ 95
  5.4.1 Component perspective .......................................... 96
  5.4.2 Interaction perspective ......................................... 99
  5.4.3 Scenario perspective ........................................... 101
  5.4.4 Indirect coupling ............................................... 104
5.5 Summary .............................................................. 107

6 Empirical Evaluation – Structural Measures 109
  6.1 Introduction ......................................................... 109
  6.2 Definition ............................................................ 109
  6.3 Planning .............................................................. 109
    6.3.1 Context .......................................................... 109
    6.3.2 Experiment Design ............................................ 110
    6.3.3 Hypotheses ..................................................... 111
    6.3.4 Variables ....................................................... 112
    6.3.5 Instrumentation ............................................... 112
  6.4 Operation ........................................................... 113
    6.4.1 Preparation .................................................... 113
    6.4.2 Execution ....................................................... 113
    6.4.3 Data validation ................................................ 114
  6.5 Analysis and Interpretation of the Results .................... 114
    6.5.1 Descriptive statistics ......................................... 114
    6.5.2 Correlation Analysis ......................................... 117
    6.5.3 Simple Regression Analysis .................................. 120
    6.5.4 Multiple Regression Analysis ................................ 123
  6.6 Threats ............................................................... 128
  6.7 Replication of Experiments ...................................... 129
    6.7.1 First Experiment Replications ............................... 129
    6.7.2 Second Experiment Replications ............................. 131
## CONTENTS

6.8 Discussion .................................................. 134
6.9 Summary .................................................... 136

7 **Empirical Evaluation - Behavioural Measures** ........................................ 137

7.1 Introduction .................................................. 137
7.2 Definition ................................................... 137
7.3 Planning ....................................................... 137
    7.3.1 Context ............................................... 137
    7.3.2 Experiment Design ................................... 138
    7.3.3 Hypotheses ........................................... 139
    7.3.4 Variables ............................................. 141
    7.3.5 Instrumentation ...................................... 141
7.4 Operation .................................................... 142
    7.4.1 Preparation .......................................... 142
    7.4.2 Execution ............................................. 142
    7.4.3 Data validation ....................................... 143
7.5 Analysis and Interpretation of the Results ........................................... 143
    7.5.1 Descriptive statistics ............................... 143
    7.5.2 Correlation Analysis ................................. 146
    7.5.3 Simple Regression Analysis .......................... 151
    7.5.4 Multiple Regression Analysis ....................... 155
    7.5.5 Threats ................................................. 165
7.6 Replication of Experiments ........................................ 166
    7.6.1 First Experiment Replications ...................... 166
    7.6.2 Second Experiment Replications ..................... 171
7.7 Discussion .................................................. 173
7.8 Summary .................................................... 177

8 **Practical Considerations** ........................................ 179

8.1 Introduction .................................................. 179
8.2 Automatic tool ............................................... 179
8.3 Product Measurement and the Production Process ................................. 181
8.4 Summary .................................................... 184
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 Conclusions</td>
<td>185</td>
</tr>
<tr>
<td>9.1 Introduction</td>
<td>185</td>
</tr>
<tr>
<td>9.2 Evaluation of the Research Objectives and Contributions</td>
<td>186</td>
</tr>
<tr>
<td>9.3 Future Research Directions</td>
<td>188</td>
</tr>
<tr>
<td>A Experimental Material – Structural Measures</td>
<td>191</td>
</tr>
<tr>
<td>B Experimental Material – Behavioural Measures</td>
<td>195</td>
</tr>
<tr>
<td>Bibliography</td>
<td>201</td>
</tr>
</tbody>
</table>
List of Figures

2.1 Software and hardware service layers in distributed systems 14
2.2 Example of the mapping of an empirical relational system to a formal relational system [Zuse, 1997] 22
2.3 ISO quality model for quality 24

3.1 Method for developing software measures 41
3.2 A software measurement abstraction: the control graph of a program 48

4.1 Remote calls among distributed components (efficiency) 63
4.2 Remote calls among distributed components (reliability) 67
4.3 Remote calls among distributed components (maintainability) 70
4.4 Sample DIS Structure 77
4.5 Sample DIS Behaviour 80

5.1 An example of the structure of a DIS 88
5.2 An example of the behaviour of a DIS 97

6.1 Scatter plot of residuals versus predicted values (TBT2) 125
6.2 Scatter plot of residuals versus predicted values (TNF2) 127

7.1 Scatter plot of residuals versus predicted values (Y = TBT2, X1 = IOS, X2 = SL) 158
7.2 Scatter plot of residuals versus predicted values (Y = TBT2, X1 = SOA, X2 = AL) 159
7.3 Scatter plot of residuals versus predicted values (Y = TNF2, X1 = IOA, X2 = AL) 161
LIST OF FIGURES

7.4 Scatter plot of residuals versus predicted values \(Y = \text{TNF}_2, X_1 = \text{SOS}, X_2 = \text{SL}\) ........................................ 162

7.5 Scatter plot of residuals versus predicted values \(Y = \text{OFC}_2\) .......................... 164

8.1 Tool for Collecting Measures from design artefacts of DIS: Collaboration
   Diagram .................................................. 181

8.2 Tool: Sample Input File ........................................... 182

8.3 Tool: Sample Output File ........................................ 182

A.1 Enterprise DIS structure ......................................... 194
List of Tables

2.1 Definitions of quality attributes (ISO 9126-1) .......................... 24
2.2 OO Design Measures proposed by Lorenz and Kidd ..................... 28
2.3 Measures proposed by Morasca [1999] ....................................... 36

4.1 Differences between the distributed case and the non-distributed case in the evaluation of the extrinsic quality of components .................... 55
4.2 The elements of the set $T$ (interaction types) .............................. 79

5.1 Structural Coupling Measurement Abstractions (Cluster Perspective) . 89
5.2 Structural Coupling Reference Abstractions (Cluster Perspective) ..... 89
5.3 Structural Coupling Measures Definition (Cluster Perspective) ........ 90
5.4 Structural Coupling Measures Legend (Cluster Perspective) .......... 90
5.5 Structural Coupling Measurement Abstractions (Component Perspective) .................................................. 91
5.8 Structural Coupling Measures Legend (Component Perspective) .... 92
5.6 Structural Coupling Reference Abstractions (Component Perspective) . 92
5.7 Structural Coupling Measures Definition (Component Perspective) .. 92
5.9 Structural Coupling Measurement Abstractions (Dependency Perspective) .................................................. 93
5.12 Structural Coupling Measures Legend (Dependency Perspective) .... 94
5.10 Structural Coupling Reference Abstractions (Dependency Perspective) . 94
5.11 Structural Coupling Measures Definition (Dependency Perspective) .. 94
5.13 Behavioural Coupling Measurement Abstractions (Component Perspective) ........................................... 98
5.14 Behavioural Coupling Reference Abstractions (Component Perspective) ............................. 98
5.15 Behavioural Coupling Measures Definition (Component Perspective) ........................................ 99
5.16 Behavioural Coupling Measures Legend (Component Perspective) .................................. 100
5.17 Behavioural Coupling Measurement Abstractions (Interaction Perspective) ........................................... 100
5.18 Behavioural Coupling Reference Abstractions (Interaction Perspective) ............................... 101
5.19 Behavioural Coupling Measures Definition (Interaction Perspective) ......................................... 101
5.20 Behavioural Coupling Measures Legend (Interaction Perspective) ......................................... 102
5.21 Behavioural Coupling Measurement Abstractions (Scenario Perspective) ................................. 103
5.22 Behavioural Coupling Reference Abstractions (Scenario Perspective) ........................................ 104
5.23 Behavioural Coupling Measures Definition (Scenario Perspective) ........................................ 104
5.24 Behavioural Coupling Measures Legend (Scenario Perspective) ........................................ 105
5.25 Behavioural Coupling Measurement Abstractions (Indirect Perspective) ....................................... 106
5.26 Behavioural Coupling Reference Abstractions (Indirect Perspective) ........................................ 106
5.27 Behavioural Coupling Measures Definition (Indirect Perspective) ........................................ 106
5.28 Behavioural Coupling Measures Legend (Indirect Perspective) ........................................ 107
5.29 Summary of the proposed measures ................................................................. 108

6.1 Experimental Goals ................................................................. 110
6.2 Structural Measures ............................................................... 112
6.3 Quality Measures ................................................................. 113
6.4 Structural Measures: Descriptive Statistics \( (n = 21) \) ......................................................... 115
6.5 Quality Measures: Descriptive Statistics \( (n = 21) \) ......................................................... 115
6.6 Structural Measures: Normality Test \( (n = 21) \) ......................................................... 116
6.7 Quality Measures: Normality Test \( (n = 21) \) ......................................................... 116
6.8 Structural measures and TBT: Correlation coefficients .................................................... 118
6.9 Structural measures and TNF: Correlation coefficients .................................................... 119
<table>
<thead>
<tr>
<th>Table Number</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.10</td>
<td>Structural measures and OFC: Correlation coefficients</td>
<td>120</td>
</tr>
<tr>
<td>6.11</td>
<td>Structural Measures and TBT: Simple Regression Statistics</td>
<td>121</td>
</tr>
<tr>
<td>6.12</td>
<td>Structural Measures and TNF: Simple Regression Statistics</td>
<td>122</td>
</tr>
<tr>
<td>6.13</td>
<td>Structural Measures and OFC: Simple Regression Statistics</td>
<td>122</td>
</tr>
<tr>
<td>6.14</td>
<td>Structural Measures and TBT: Simple Regression Statistics</td>
<td>124</td>
</tr>
<tr>
<td>6.15</td>
<td>Independence of the residuals (TBT)</td>
<td>124</td>
</tr>
<tr>
<td>6.16</td>
<td>Normality of the residuals (TBT)</td>
<td>125</td>
</tr>
<tr>
<td>6.17</td>
<td>Structural Measures and TNF: Multiple Regression Statistics</td>
<td>126</td>
</tr>
<tr>
<td>6.18</td>
<td>Independence of the residuals (TNF)</td>
<td>127</td>
</tr>
<tr>
<td>6.19</td>
<td>Normality of the residuals (TNF)</td>
<td>128</td>
</tr>
<tr>
<td>6.20</td>
<td>TBT: Descriptive Statistics ($n = 21$)</td>
<td>130</td>
</tr>
<tr>
<td>6.21</td>
<td>TBT: Normality Test ($n = 21$)</td>
<td>130</td>
</tr>
<tr>
<td>6.22</td>
<td>Structural measures and TBT: Spearman's Correlation coefficients (Replications)</td>
<td>130</td>
</tr>
<tr>
<td>6.23</td>
<td>Structural Measures and TBT: Simple Regression Statistics (Replications)</td>
<td>131</td>
</tr>
<tr>
<td>6.24</td>
<td>Structural Measures and TBT: Multiple Regression Statistics (Replications)</td>
<td>132</td>
</tr>
<tr>
<td>6.25</td>
<td>TNF: Descriptive Statistics ($n = 21$)</td>
<td>132</td>
</tr>
<tr>
<td>6.26</td>
<td>TNF: Normality Test ($n = 21$)</td>
<td>132</td>
</tr>
<tr>
<td>6.27</td>
<td>Structural measures and TNF: Spearman's correlation coefficients (Replications)</td>
<td>133</td>
</tr>
<tr>
<td>6.28</td>
<td>Structural Measures and TNF: Simple Regression Statistics (Replications)</td>
<td>133</td>
</tr>
<tr>
<td>6.29</td>
<td>Structural Measures and TNF: Multiple Regression Statistics (Replications)</td>
<td>134</td>
</tr>
<tr>
<td>6.30</td>
<td>Summary of relationships among structural measures and quality attributes</td>
<td>135</td>
</tr>
<tr>
<td>7.1</td>
<td>Experimental Goals</td>
<td>138</td>
</tr>
<tr>
<td>7.2</td>
<td>Behavioural Measures</td>
<td>141</td>
</tr>
<tr>
<td>7.3</td>
<td>Quality Measures</td>
<td>142</td>
</tr>
</tbody>
</table>
LIST OF TABLES

7.4 Behavioural Measures: Descriptive Statistics \( (n = 21) \) ........................................ 144
7.5 Quality Measures: Descriptive Statistics \( (n = 21) \) ........................................ 145
7.6 Behavioural Measures: Normality Test \( (n = 21) \) ........................................ 145
7.7 Quality Measures: Normality Test \( (n = 21) \) ........................................ 146
7.8 Behavioural measures and TBT: Correlation coefficients ........................................ 148
7.9 Behavioural measures and TNF: Correlation coefficients ........................................ 150
7.10 Behavioural measures and OFC: Correlation coefficients ...................................... 151
7.11 Behavioural Measures and TBT: Simple Regression Statistics ................................... 153
7.12 Behavioural Measures and TNF: Simple Regression Statistics ................................... 154
7.13 Behavioural Measures and OFC: Simple Regression Statistics .................................. 155
7.14 Behavioural Measures and TBT: Multiple Regression Statistics ................................ 156
7.15 Independence of the residuals (TBT): Durbin-Watson Test .................................... 157
7.16 Normality of the residuals (TBT): Shapiro-Wilk Test ........................................... 157
7.17 Behavioural Measures and TNF: Multiple Regression Statistics .............................. 160
7.18 Independence of the residuals (TNF): Durbin-Watson Test .................................... 162
7.19 Normality of the residuals (TNF): Shapiro-Wilk Test ........................................... 162
7.20 Behavioural Measures and OFC: Multiple Regression Statistics ............................. 163
7.21 Independence of the residuals (OFC) ................................................................. 164
7.22 Normality of the residuals (OFC): Shapiro-Wilk Test ........................................... 165
7.23 TBT: Descriptive Statistics \( (n = 21) \) ................................................................. 167
7.24 TBT: Normality Test \( (n = 21) \) ................................................................. 167
7.25 Behavioural measures and TBT: Spearman's Correlation coefficients (Replications) .......... 168
7.26 Behavioural Measures and TBT: Simple Regression Statistics (Replications) ............... 169
7.27 Behavioural Measures and TBT: Multiple Regression Statistics (Replications) ............. 170
7.28 TNF: Descriptive Statistics \( (n = 21) \) ................................................................. 171
7.29 TNF: Normality Test \( (n = 21) \) ................................................................. 171
7.30 Behavioural measures and TNF: Spearman's correlation coefficients (Replications) .......... 172
LIST OF TABLES

7.31 Behavioural Measures and TNF: Simple Regression Statistics (Replications) ........................................ 174
7.32 Behavioural Measures and TNF: Multiple Regression Statistics (Replications) ........................................ 175
7.33 Summary of relationships among behavioural measures and quality attributes ........................................ 176

A.1 Values of the Structural Measures (Component and Cluster Perspective) 192
A.2 Values of the Structural Measures (Dependency Perspective) ............ 193

B.1 Values of the Behavioural Measures (Component Perspective) ............ 196
B.2 Values of the Behavioural Measures (Interaction perspective) .............. 197
B.3 Values of the Behavioural Measures (Scenario Perspective) ............... 198
B.4 Values of the Behavioural Measures (Indirect perspective) ............... 199

xix
Summary

Enterprise information systems are increasingly being developed as distributed information systems. Quality attributes of distributed information systems, as in the centralised case, should be evaluated as early and as accurately as possible in the software engineering process. In particular, software measures associated with quality attributes of such systems should consider the characteristics of modern distributed technologies.

Early design decisions have a deep impact on the implementation of distributed enterprise information systems and thus, on the ultimate quality of the software as an operational entity. Due to the fact that the distributed-software engineering process affords software engineers a number of design alternatives, it is important to develop tools and guidelines that can be used to assess and compare design artefacts quantitatively.

This dissertation makes a contribution to the field of Software Engineering by proposing and evaluating software design measures for distributed enterprise information systems. In previous research, measures developed for distributed software have been focused in code attributes, and thus, only provide feedback towards the end of the software engineering process. In contrast, this thesis proposes a number of specific design measures that provide quantitative information before the implementation. These measures capture attributes of the structure and behaviour of distributed information systems that are deemed important to assess their quality attributes, based on the analysis of the problem domain.

The measures were evaluated theoretically and empirically as part of a well-defined methodology. On the one hand, we have followed a formal framework based on the theory of measurement, in order to carry out the theoretical validation of the
proposed measures. On the other hand, the suitability of the measures, to be used as indicators of quality attributes, was evaluated empirically with a robust statistical technique for exploratory research. The data sets analysed were gathered after running several experiments and replications with a distributed enterprise information system. The results of the empirical evaluation show that most of the proposed measures are correlated to the quality attributes of interest, and that most of these measures may be used, individually or in combination, for the estimation of these quality attributes—namely efficiency, reliability and maintainability.

The design of a distributed information system is modelled as a combination of its structure, which reflects static characteristics, and its behaviour, which captures complementary dynamic aspects. The behavioural measures showed slightly better individual and combined results than the structural measures in the experimentation. This was in line with our expectations, since the measures were evaluated as indicators of non-functional quality attributes of the operational system. On the other hand, the structural measures provide useful feedback that is available earlier in the software engineering process.

Finally, we developed a prototype application to collect the proposed measures automatically and examined typical real-world scenarios where the measures may be used to make design decisions as part of the software engineering process.
Chapter 1

Introduction

1.1 Motivation

Information systems have become an important part of the daily operations of enterprises. Therefore, the construction of high quality information systems is one of the most pressing challenges for enterprises today. The design of an information system is a major determinant of the quality of the overall system. Furthermore, the more effort is spent evaluating the design of an information system, the more likely the effort saving in later phases of the development life-cycle. Finally, one of the most effective ways of assessing the design of an information system is by using specific measures developed for this purpose.

1.1.1 Software

Software engineers construct non-physical, abstract artefacts, unlike traditional engineers that construct physical artefacts. Software does not obey the laws of physics like buildings and cars do [Henderson, 2003]. Furthermore, some of the intuitive perceptions that people have for physical artefacts are not present when they relate to software. A large software system is characterised by a large number of abstract characteristics, making it difficult to understand and appreciate. In a few words, software is complex.

Software is also costly. Annual government spending on software in the US alone is estimated at $22.5 billion [Applewhite, 2003], while US corporations spend
more than $250 billion each year on software development projects [The Standish Group, 1999]. Japanese companies invest around 15 percent of their total budget on software [Matsubara, 2001]. In Europe, 42 percent of the software projects cost more than one million of euros [EU, 2001].

Despite its complexity and cost, software has become essential to individuals and organisations. Over the last 40 years software has evolved from a specialised problem solving and information management tool to an industry in itself. A software-driven Internet has spawned an estimated world-wide market of US$ 6.8 trillion, according statistics reported by the European Union [Hobley, 2001]. Software is not only restricted to the world of computers. A wide variety of products today contain embedded software—such as PDAs, mobile telephones, DVD players, cars and aeroplanes—and in the future this trend is set to continue. The market for these products is forecasted to grow exponentially in the next 10 years [Graaf et al., 2003].

Unfortunately, the software industry track record has been far from impressive. Gross measures presented in the literature indicate that software productivity has been dropping more rapidly than many others industries, and the software industry had one of the worst productivity declines (10%) from 1990 to 1995. In that same period, productivity for the hardware industry grew by 86 percent [Anselmo and Ledgard, 2003].

Over the years, many industry observers have suggested that software is constantly associated with late deliveries, high costs and poor quality [Boehm, 1981; DeMarco, 1982; Pfleeger, 2001; Sommerville, 2001]. Some of the (US) statistics reported are disappointing [The Standish Group, 2001]:

- Only 28 percent of all software projects are on time and within budget and have all their planned functionality;

- 23 percent of software projects are abandoned;

- Time overruns are estimated to be 63 percent over the original time;

- Cost overruns average 45 percent over the original estimate;
Only 67 percent of the required functionality is delivered with the final product.

In today's competitive, time-to-market driven environments, software engineers are faced with increasing demands for better productivity and higher quality from their enterprises. However, cost-effective software of high quality is hard to achieve.

1.1.2 Engineering software

Software engineers and managers frequently adopt new technological solutions to improve software development within their enterprises, often with mixed results. While some technologies have delivered gains in terms of quality and timeliness of software delivery, certain technologies have had little, or even negative, impact.

This is due to the fact that technology alone does not guarantee substantial improvement; rather, improvement is the combination of technology and several additional factors such as proper use of the technologies, people training and process (re)engineering [Duchessi and Chengalur-Smith, 1998]. Although appropriate deployment of new technologies may be necessary, it may not be sufficient to succeed. Process improvement and organisational changes—among others—are also required [Paulish and Carleton, 1994].

Leading edge enterprises have understood this, and over the years have worked to improve their software development processes. For example, following standards such as CMM, SPICE or ISO 9000, they are attempting to transform the craft nature of software development into an engineering production process.

During this course of action, a significant challenge to software engineers is to avoid neglecting the software process while advancing along the technology dimension. To be successful, they need to be able to advance simultaneously along these two dimensions. Measurement can be helpful to handle this challenge, since it offers visibility into the ways in which processes, products, methods and technologies relate to one another [Fenton and Pfleeger, 1997].

Therefore, this dissertation intends to contribute towards improving the software production process when using new distributed technologies.
1.1.2.1 Engineering enterprise Distributed Information Systems (DIS)

New technologies, such as CORBA, J2EE, .NET and Web Services, promise to alleviate some of the problems of enterprise software development. Major software vendors are aggressively marketing products with distributed computing capabilities\(^1\). Most enterprises now use some commercial technology with distributed capabilities to run their businesses. Key drivers behind this trend include increasing Internet usage and the ongoing need to integrate heterogeneous legacy systems [Liu and Gorton, 2003].

The proliferation of these new technologies presents several challenges to the Software Engineering community. Arguably, software engineering processes for DIS have not accompanied the fast growth of distributed technologies, and the practice of DIS development has been often technology driven, rather than principle driven.

Consequently, there is a growing concern in the Software Engineering community about how DIS should be designed and implemented. In particular, there is a growing interest in the evaluation of these software artefacts. Without appropriate processes and tools, it will be harder to manage costs and develop enterprise DIS within a reasonable time [Schmidt et al., 2000]. The development of enterprise DIS by means of ad hoc strategies has a lower probability of success in the implementation and maintenance stages [Wijegunaratne and Fernandez, 1998]. To avoid unnecessary risks in the development of DIS, it is imperative to employ disciplined methods to produce and evaluate these software artefacts, and to use tools that are robust and flexible. These processes and tools should take into account specific aspects of the distributed scenario, such as:

- the multiplicity and/or heterogeneity of hardware and software inherent to DIS;
- the possibility of parallelism and/or concurrency of executing components of a DIS;
- the importance of the supporting middleware infrastructure for the develop-

\(^{1}\)For example Borland, IBM, Microsoft, Oracle and Sun.
ment and the execution of enterprise DIS;

- the effect on quality of the different possible communication modes among DIS components;

- the impact of middleware types and middleware interfaces in the design and implementation of DIS.

Principally, it is necessary to have an engineering approach for the production and evaluation of enterprise DIS. That is, the establishment and use of processes and tools based on principles and practices based on research and industrial experience, recognised by the Software Engineering community. Although the use of processes and tools established upon engineering principles and practices will not provide us with the "silver bullet" [Brooks, 1987], it may help to minimise problems and risks, and provide the groundwork to manage effectively the software development process and the evaluation of (intermediate and final) software products.

Furthermore, we argue that the systematic use of Software Engineering methods, models, tools and techniques for the production of quality software artefacts should be common practice in DIS projects. In this sense, a main objective of this work is to contribute with a measurement approach towards the evaluation and comparison of software artefacts produced by the development process of DIS. Some aspects relevant to the software development process for enterprise DIS, such as architectural patterns [Schmidt et al., 2000; Fowler, 2003], have been discussed in the literature. However, other aspects such as measurement, have received much less attention.

1.1.3 Measurement in software engineering

Drucker [1988] points out that measurement is an essential function of management. He also emphasises that measurement is crucial to guide the behaviour and performance of an organisation. Not far from our domain, engineers commonly measure variables such as network down-time and CPU usage to manage hardware operations. Unfortunately, measurement is less common in the area of
software engineering. Oman and Pfleger [1997] give some examples of why measurement is important in Software Engineering:

- To understand and evaluate the factors that affect the quality of software products and processes.
- To manage risks and control projects.
- To monitor and improve the software production process and their products.
- To plan projects and estimate variables such as time, cost and development effort.
- To conduct experiments and case studies effectively.

Measurement is gradually becoming more important in the practice of software engineering, and measurement research has grown substantially from early contributions such as the cyclomatic number [McCabe, 1976] and function points [Albrecht and Gaffney, 1983]. However, software measurement is a young discipline still and requires continuous research attention to mature.

1.1.3.1 The importance of DIS Measurement

The focus on process improvement and quality control has increased the demand for software measures to better manage the development of software. The need for measures is particularly crucial in the case of an enterprise adopting new (distributed) technologies, for which standard practices have yet to be established.

Due to the fundamental differences between traditional centralised systems and modern distributed systems, several authors argue that the production of software for distributed environments may even require different capabilities in the development process [Fernandez and Wijegunaratne, 1998; Emmerich, 2000b].

Given this contrast, it is not surprising that software measures developed for traditional centralised information systems cannot readily be accommodated to reflect DIS unique aspects—discussed in Chapter 2.

The need for new measures especially developed for distributed software has been suggested by a number of researchers—for example Shatz [1988] and Cheng
[1993]. Therefore, given that traditional measures may be perceived as not supporting key DIS aspects, it seems appropriate to attempt to develop a set of new measures especially designed to cater for the singular features of DIS.

1.2 Overview of this dissertation

The motivation for this dissertation arose from the recognition that the development of enterprise software—in particular for distributed environments—is an increasingly important activity. The engineering of this kind of software (i.e. enterprise DIS) poses several open questions to researchers and practitioners.

In the past, Software Engineering has had a tradition of being a black art, with the perception that typical software engineers could do little to manage their projects. Although there is a need for improvement still, many researchers contend that this unique activity can be proactively managed with a better measurement of the software production process and its products [Pressman, 2001; Pfeeger, 2001].

An overriding goal of Software Engineering is to produce high-quality software products, and it is widely acknowledged that measurement of the design is fundamental to estimate early the quality of the final product. However, for measurement to take place, it is necessary to study the relationships between design attributes and final product quality. Due to the fact that quality is really a composite of many attributes, we intend to focus on a subset of especially important attributes, based on an analysis of the problem domain.

Research in software measurement follows two broad streams, one that focuses on practical relevance and another one that focuses on formalistic rigour. These streams are not mutually exclusive and several researchers have tried to bridge the two streams following an integral approach. This dissertation follows such an approach to measure software attributes of DIS artefacts.

1.2.1 Objectives of this dissertation

The main objective of this dissertation is to define a suite of software design measures to be used as early indicators of quality attributes for enterprise DIS. The main objective has been refined into a set of specific sub-objectives:
CHAPTER 1. INTRODUCTION

- Analyse critically the state of the art of measurement for software products, in particular for DIS artefacts.

- Approach the measurement of software artefacts with a well-defined methodology.

- Define theoretically valid software design measures, based on practical goals and hypotheses that are consistent with the intuitive analysis of the problem domain.

- Evaluate the empirical validity and practical usefulness of the measures as indicators of quality attributes.

- Design and construct a tool for the automatic collection of the software measures

- Discuss practical ways in which these measures might be used in the software engineering process of real projects.

1.2.2 Outline of the rest of the dissertation

The next chapter presents the background to this investigation and related work. The research method is described in Chapter 3. Chapter 4 presents the informal and formal measurement models. The formal definition and theoretical validation of the measures is put forward in Chapter 5. Chapter 6 and 7 focus on the empirical validation of the measures. Chapter 8 describes the practical role the proposed measures might play in the software engineering process. Finally, the last chapter presents concluding remarks and future research lines.
Chapter 2

Background and Related Work

2.1 Introduction

The objective of this research, as stated in the previous chapter, is to develop a suite of software design measures for enterprise DIS. To this end, this chapter reviews relevant background in the areas of software measurement and distributed enterprise information systems. Section 2.2 discusses the characteristics of DIS, and explains specific issues faced by the software engineer when designing DIS. A review of software measurement is essential for understanding precisely the terminology and the context of this work, hence, this is the purpose of section 2.3. Finally, section 2.4 presents a critical analysis of the state of the art of software measures for DIS artefacts.

2.2 Distributed Information Systems

The term distributed computing is often used in the literature with different meanings. Hence, it is important to clarify how this concept is used in this dissertation. Informally, a distributed system may be characterised as a collection of independent computers linked by a network [Emmerich, 2000a]. Accordingly, a key issue is to consider how the network of computers is used by software applications:

- The network can be used to support the execution of independent jobs that arrive randomly at various computers. The major role of the network, in
this case, is the provision for resource sharing. This type of processing has been called network computing [Shatz, 1993]. In network computing, jobs are dynamic, and computers communicate mainly by transferring files.

- The network can be used to support the execution of autonomous components (or processes) that cooperate with one another to achieve common goals. This type of processing has been called cooperative computing [Shatz, 1993]. In cooperative computing, processes are static and communicate by making use of the programming interfaces of the supporting middleware.

In this dissertation we take the view that a distributed system is comprised of a fixed set of components, the structure of which has been defined in the design phase. Therefore, in this thesis, the term “distributed computing” may be closely associated to the term “cooperative computing”.

We have argued before that modern distributed systems are different from traditional centralised systems. A comparison between a distributed system with a typical centralised system shows key distinctive aspects as follows [Emmerich, 2000a]:

- A centralised system is typically composed of different parts. However, these parts, such as the objects of a program, are not autonomous. The system possesses full control over its parts at any one time. By contrast, in a distributed system often there is no master component that possesses control over all other components. That is, components are autonomous.

- Centralised systems are mostly homogeneous. They tend to be constructed using the same infrastructure, and the same programming language is used for all components in most cases. On the contrary, the components of a distributed system need not be homogeneous. In fact, they are often heterogeneous. This heterogeneity applies to all of the following: computer hardware, networks, operating systems, middleware and programming languages.

- A centralised system is very likely to be built to run as a single process, even to construct it single-threaded. Furthermore, components are usually executed sequentially on the same computer. In contrast, as a consequence of component autonomy, distributed systems commonly execute components
concurrently. In addition, components are often multi-threaded and generally they are not executed on the same computer.

- Most failures of a centralised system are total. That is, when a failure occurs, the whole system fails. For example, a power outage or an operating system crash may cause the unavailability of an entire centralised system. Conversely, almost all failures of a distributed system are partial. Hence, in many occasions, components of a distributed system are fully operational when other components do not function properly or do not function at all.

We believe these differences impact significantly on the evaluation of quality attributes such as efficiency, reliability and maintainability. Therefore, it is necessary to revise, adapt and extend established Software Engineering methods and measures thoroughly before being able to apply them to the domain of DIS. This is also argued in the domain of web information systems, where several researchers state that traditional methods and measures should be reviewed and re-assessed before they can be applied to this domain [Baresi et al., 2003; Mendes et al., 2003; Ruhe et al., 2003].

2.2.1 Middleware

Middleware is an additional layer of software used to construct modern distributed enterprise information systems. Figure 2.1 depicts the position of middleware in the layered technological architecture of enterprise DIS.

Its purpose is to mask as much as possible the heterogeneity of the underlying collection of computing platforms, and to provide as much as possible a homogeneous infrastructure and appropriate interfaces to construct distributed software applications [Tanenbaum and van Steen, 2002].

Middleware provides the software engineer with the possibility of accessing the same collection of services (via their interfaces), regardless of the underlying operating system and hardware [Coulouris et al., 2001]. For example:

- JDBC [Sun, 2003a], Java Data-Base Connectivity, provides a set of classes that allows a Java components to interact with several different databases
with the same API (Application Programming Interface);

- MQ-Series [IBM, 2003] provides a generic interface to access message queues that enables heterogeneous components to interchange information asynchronously and reliably.

At the highest level—i.e. application level, see Figure 2.1—inter-component communication in a DIS is based ultimately on the low-level communication facilities offered by the underlying network. Expressing communication through network primitives such as sockets is harder than using primitives based on shared memory, such as a procedure call or a method invocation. Unless the primitive communication facilities of a computer network are replaced by high-level primitives, development of enterprise DIS is extremely difficult. Schmidt et al. [2000] enumerate some of these complexities:

- Excessive low-level details
- Rediscovery and reinvention of incompatible high-level programming abstractions
- High-potential for errors
- Lack of portability
2.2. DISTRIBUTED INFORMATION SYSTEMS

- Steep learning curve
- Inability to scale up to handle growth

To this we can add lack of available expertise in the enterprise and hence difficulty of maintenance.

Middleware software has been devised in order to conceal these difficulties from software engineers as much as possible. Therefore, as they solve real problems and simplify DIS construction, middleware solutions are rapidly being adopted in the enterprise world [Charles, 1999].

Most middleware is based on some model for describing distribution and communication to make development and integration of DIS less complex. A relatively simple model—followed by the World Wide Web—is that of treating everything as a file. Another important early middleware model is based on a Remote Procedure Call (RPC). This is an extension of the notion of a local procedure call. In this model, the emphasis is on allowing a procedure to call another procedure whose implementation is located on a remote computer. RPC was the starting point for the notion of distributed objects and object-oriented middleware, since enabling procedure calls across computer boundaries made also possible to invoke remote object methods. In order to support proper data storage and management, middleware may offer facilities for distributed transactions and SQL-based communication. Message-oriented middleware is characterised by the capability of providing asynchronous and reliable interchange of information. The growing interest in using XML as a vehicle for remote communication is driving the widespread acceptance of technologies such as Web services.

If we assume that a middleware infrastructure is used for the development of application software, research into software engineering for enterprise DIS must deliver principles, guidelines, methods and tools that are compatible with and enable use of the capabilities that current middleware products provide to achieve real industrial significance [Emmerich, 2000b]. Middleware research, however, mainly discusses implementation issues and tends to ignore other earlier activities that are part of the DIS software engineering processes. With only few exceptions [Di Nito and Rosenblum, 1999; Dashofy et al., 1999], there is little work on the influence of
middleware on software design of DIS.

We believe the use of middleware should not be entirely transparent to the software engineer before the implementation stage. In the design stage, the DIS software engineer needs to be aware of the role of middleware in the communication between distributed components, and to examine carefully the available options for components on different machines to interact.

### 2.2.2 Enterprise Information Systems

Hitherto, we have discussed the characteristics of enterprise DIS mainly from the infrastructure perspective. In this section, we describe enterprise DIS from the application perspective.

A large amount of computer software is produced. However, there are distinct kinds of existing software, each with its own challenges and difficulties. In this thesis we have concentrated on enterprise applications. Although a precise definition of enterprise application may be difficult to provide, it is possible to give a clear indication of its meaning in this context.

Examples of enterprise applications include insurance claims processing, payroll, patient records, customer service, supply chain, and shipping tracking. Enterprise applications do not include word processors, games, chemical plant controllers, telephone switches, operating systems and compilers.

The most important characterisation in our context is that enterprise applications rarely live isolated. The imperatives of application integration usually impose the need to interact with other applications distributed around the enterprise. The various applications may have been built at different times with different technologies, and even the integration mechanism may be different. Even if an enterprise integrates all applications with a common technology, it will be faced with the problem of disperse data in heterogeneous syntactic and semantic formats.

Enterprise information systems are frequently distributed because they tend to reflect the structure of organizations and the way in which organizational units interact with each other. In these cases, the components of a system that have been developed independently and operate autonomously are made to exchange
information and/or processing.

We consider the components of such DIS as the fundamental units to be studied. However, components are very often naturally grouped together, because they are part of the same subsystem, or because they execute on the same hardware platform. Hence, we will also consider these clusters of such components.

Even though we are particularly interested in enterprise information systems, the generality of our approach may be suitable to a broader range of DIS.

### 2.2.3 Software Engineering with Distributed Technologies

Over the last few years, a great number of distributed computing products and technologies emerged and gained popularity. Unfortunately, these technologies do not point to a single direction for the software engineer—rather they have opened the door to several competing forms of software structure and behaviour. In addition, the appearance of principles and methods for putting distributed technologies to "good" use has been usually slower than the take up of the technologies themselves [Fernandez and Wijegunaratne, 1998].

The typical focus in the research literature is on capabilities of distributed computing infrastructure products, networking and connectivity implementation issues. This focus is important and necessary, but we argue it is not sufficient to achieve the desired quality outcomes of an enterprise DIS project. Proper design of the software, built upon an appropriate infrastructure, is what really can make the difference.

It is possible to extend or adapt principles from the centralised scenario, but it is crucially important to recognise that a DIS need not be implemented as a hierarchical set of components controlled by a root component, as it is typical of the centralised case.

Furthermore, it is necessary to recognise the fundamental differences between the engineering of distributed software components housed on remote computers and the engineering of local components that reside in a centralised system. We consider that those differences should be taken into account as early as possible in the software development life-cycle, that is, before the implementation stage.
It is widely accepted that maintainability of the software is affected by the software design. However, other quality attributes such as efficiency and reliability seem to have been tied to the software implementation, leaving their consideration out of the design. While this approach may be suitable for centralised components, it may not be appropriate in the case of distributed components [Waldo et al., 1997].

Emmerich [2000a] discusses the major differences when designing local and distributed software components:

- **Latency.** The difference in performance between a local and a remote call is currently, on average, of many orders of magnitude. Given the relative rates at which computer speed and network speed are changing, the difference in the future is likely to worsen. Hence, although the overhead for a local call may be negligible, the overhead of remote calls is likely to be very significant and cannot not be ignored.

- **Component references.** References to local components are implemented through memory addresses (or pointers), which are light-weight structures. References to distributed components are more substantial data structures, since they need to encode additional information such as location details—for example, a simple location mechanism is the combination of IP host address and TCP port number.

- **Communication.** In centralised systems, there is essentially one way for two local components to interact: a procedure or method call. In this case the call is between two peers, the called component is assumed to be available, and the caller component is likely to block waiting for the result. However, different ways of interaction among distributed components are possible due to different existing modes of synchronisation, availability and multiplicity.

- **Parallelism.** Local components are usually executed sequentially. Although it is possible to achieve logical concurrency, only one component really is executing at any given point in time. On the other hand, distributed components are often logically and physically executing in parallel. In addition, if concurrency is not controlled properly, the integrity of components may be at risk.
2.3 Software Measurement

The conceptual framework of Fenton [1994] for software measurement is well accepted in the software engineering community, and we believe it provides adequate support for this research. This framework has three dimensions.

It suggests that a key task of any software measurement is to identify the entities of interest that we wish to measure, so the first dimension classifies software entities. There are three classes of entity we may intend to measure:

- a process: a collection of related software engineering activities (e.g. a programming technique)
- a product: an artefact that is the output of the process activities (e.g. the code of a program)
- a resource: an input used by the process activities to produce artefacts (e.g. a programmer)

Any aspect we measure of these entities is an attribute, so the second dimension focuses on software attributes. Attributes are classified as either internal or external. Internal attributes of an entity are those that can be measured purely in terms of the entity itself, such as the size and cohesion of a sub-program. External attributes of an entity are those that can only be measured with respect to how
the entity relates to its environment, for example the usability and portability of a program.

The third dimension of the framework is about the type of measurement. On the one hand, the attributes of entities are measured directly or indirectly. Direct measurement of an attribute is a measurement that does not depend on the measurement of any other attribute, such as a program's length in terms of lines of code. Indirect measurement of an attribute is a measurement that involves the measurement of one or more other attributes, for instance a program's average modularity in terms of lines of code per sub-program. On the other hand, the attributes of entities can be measured objectively or subjectively. The measurement is objective if performed by different people always produces the same result, such as counting the number of methods of a class in an OO program. Subjective measurement can produce different results when executed by different persons. For instance, the understandability of an OO program rated from 1 to 10 is a subjective measure.

Finally, Fenton [1994] also states that there are two broad uses of software measurement: assessment and prediction. Measurement for assessment is helpful in understanding existing entities. Nevertheless, in many circumstances, we need to predict an attribute of some entity that does not exist yet. Predictive measurement of an attribute normally depends on a model relating this attribute to some existing measures of other attributes—although the model need not be complex to be useful. For example, measurement of attributes of early artefacts (such as complexity of the design) can be used to predict measures of attributes of later artefacts (such as the maintainability of the code). Thus, precise predictive measurement of an attribute is unavoidably dependent on cautious assessment measurement of other attributes. However, for predictive measurement the model by itself is not always enough, and procedures for interpreting the results (and possibly for determining the model parameters) may be necessary.

According to this framework, our research is focused on the measurement of software products for the assessment of internal attributes and the prediction of external attributes.
2.3.1 Measurement Theory

A detailed and formal approach is desirable to drive the measurement of software product attributes. Fortunately, there are a number of theories that may be used to guide the measurement of software products. One of these is the representational theory of measurement. This theory, often simply called measurement theory, offers a number of prescriptions regarding measurement [Roberts, 1979].

The underlying notion of measurement theory is that if in our problem domain or universe of discourse (UoD) there exists an intuitive or empirical understanding of relationships of objects within that UoD, then those relationships may be formalised mathematically. In addition, we might have some common understanding of one or more binary operations on the objects of our UoD.

This “qualitative” system can be written formally as follows: an empirical relational system, $E$, is an ordered tuple $E = (A, r_1, \ldots, r_n, o_1, \ldots, o_m)$, where $A$ is a set of objects, $r_1, \ldots, r_n$ are empirical relations, and $o_1, \ldots, o_m$ are binary operations on the empirical objects in the set $A$. To be able to measure some characteristic about an object, an empirical relational system $E$ needs to be mapped to a formal relational system $F$.

This formal relational system is defined as $F = (B, r'_1, \ldots, r'_n, o'_1, \ldots, o'_m)$, where $B$ is a set of formal objects, $r'_1, \ldots, r'_n$ are relations on those objects in $B$, and $o'_1, \ldots, o'_m$ are the binary operations.

The required mapping (measure) is accomplished by a function $\mu : E \rightarrow F$, which yields for every empirical object $a \in A$ a formal object $\mu(a) = b \in B$. This mapping may not be arbitrary. It is a mapping that preserves all relationships and operations. A mapping from one relational system to another that preserves all relations and operations is a homomorphism [Roberts, 1979].

The example of Figure 2.2 illustrates the mapping between an empirical relational system and a formal relational system by considering the measurement of length of a physical object.

The empirical relation “board $a$ is longer than board $a'$” in the example is mapped to the formal relation “real number $b >$ real number $b'$”, enabling the explicit understanding of the length of wooden boards. It is important to note that we can
decide whether one board is longer than another without measurement. That is, there is a clear intuitive idea of length independently from measurement. However, software artefacts are not physical objects, and their relations are not so well understood generally. Therefore, while a mapping may seem unnecessary for the example above, it is helpful to understand some software attributes.

### 2.3.2 Software Product Quality

In the debate about the meaning of software quality, the definitions of software quality range from quality considered as synonymous of ‘excellence’ to quality understood as the degree of compliance with specified requirements. Quality is also an ambiguous and multidimensional concept; therefore it is inevitable that different views exist [Gillies, 1992]. These views are often diverse, and they may even conflict with each other. A distinction is commonly made between the user and the engineer view of quality. For instance, the user is concerned primarily by the quality attributes of the operational software, while the engineer is also interested on the quality attributes of the source code. The spectrum can be broader if we consider other perspectives within a software project such as the manager.

In order to evaluate software quality and to compare software quality in different situations, many models of software quality have been proposed. Most of them are hierarchical in nature, and hence they are structured top-down. The upper branches hold important high-level attributes of software quality and each attribute is decomposed into lower-level sub-attributes. For example, the attribute maintainability may be decomposed into the sub-attributes simplicity, modular-
ity and self-descriptiveness [IEEE, 1992]. The sub-attributes are easier to assess and measure than the attributes; therefore actual measures are proposed for the sub-attributes. A tree structure describes the relationship between the attributes and the sub-attributes, thus the attributes can be measured indirectly in terms of dependent sub-attributes measures.

Although the models use different approaches and terminology making difficult to understand exactly what quality is, still they are useful in articulating what people think is important and of general interest. The early models include the work of McCall et al. [1977], Boehm et al. [1978], Grady and Caswell [1987], and Watts [1987]. Recently, the IEEE [1998] and ISO [2001] have tried to standardise a model for software quality.

The IEEE standard 1061-98 [1998] provides a flexible model designed to permit additions, deletions, and modifications of quality attributes and quality sub-attributes\(^1\). The ISO/IEC standard 9126-1 [2001] goes further and describes a prescriptive model for software product quality.

The ISO model is followed in this dissertation to refer to quality attributes and sub-attributes. This model specifies six attributes for quality, which are further subdivided into sub-attributes\(^2\)—see Figure 2.3. Table 2.1 provides definitions for the quality attributes. The quality sub-attributes relevant to this dissertation will be defined after the measurement goals and hypotheses have been established.

### 2.3.3 Software Design Measures

This section surveys briefly some of the well-known measures encountered for traditional non-distributed software, which illustrate how design attributes are characterised and quantified for centralised software.

#### 2.3.3.1 Information Flow Measures

Henry and Kafura [1981] define some measures based on the information flow between system components. Specific measures are defined for module complexity

---

\(^1\)This (IEEE) standard use the terms factor and sub-factor in reference to attributes and sub-attributes.

\(^2\)This (ISO/IEC) standard use the terms characteristic and sub-characteristic in reference to attributes and sub-attributes.
### Figure 2.3: ISO quality model for quality

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Efficiency</td>
<td>The capability of the software product to provide appropriate performance, relative to the amount of resources used, under stated conditions</td>
</tr>
<tr>
<td>Functionality</td>
<td>The capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions</td>
</tr>
<tr>
<td>Maintainability</td>
<td>The capability of the software product to be modified. Modifications may include corrections, improvements or adaptation of the software to changes in environment, and in requirements and functional specifications</td>
</tr>
<tr>
<td>Portability</td>
<td>The capability of the software product to be transferred from one environment to another</td>
</tr>
<tr>
<td>Reliability</td>
<td>The capability of the software product to maintain a specified level of performance when used under specified conditions</td>
</tr>
<tr>
<td>Usability</td>
<td>The capability of the software product to be understood, learned, used and attractive to the user, when used under specified conditions.</td>
</tr>
</tbody>
</table>

Table 2.1: Definitions of quality attributes (ISO 9126-1)
and sub-system complexity. These measures are evaluated empirically using the source code of the UNIX operating system.

The authors state that there is a global information flow from module A to module B through a global structure D if A writes information into D and B reads information from D, and that there is local information flow from module A to module B if one or more of the following conditions hold:

- if A calls B,
- if B calls A and A returns a value to B, which B utilises, or
- if C calls both A and B passing a value from A to B.

The fan-in of a module A is defined as the number of local flows into module A plus the number of data structures from which module A reads information. The fan-out of a module A is the number of local flows from module A plus the number of data structures to which module A writes information.

The presented complexity measure defines a value that attempts to measure the "degree of simplicity" of relationships between modules. They state that the complexity of a module depends on two factors: the complexity of the module code and the complexity of the module connections to its environment. A very simple length measure was used as an index of module code complexity: the length of a module was defined as the number of lines of text in the source code for the module. The complexity of module connections to its environment is determined by the fan-in and fan-out values. The complexity measure of a module is proposed as $\text{length} \times (\text{fan-in} \times \text{fan-out})^2$ in the first instance. The weighting of the second part is based on the belief of the authors that complexity is more than linear in terms of the connections that a module has to its environment. The authors afterwards suggest that the first part, since it is only a weak factor, may be omitted without significant loss of accuracy. Thus, the measure is simplified to $(\text{fan-in} \times \text{fan-out})^2$.

A sub-system with respect of a data structure D consists of those modules that either directly write information into D or directly read information from D. The complexity of a sub-system is defined to be the sum of the complexities of the
modules within the sub-system. The experimental results show that most of a
sub-system's complexity (about 85%) is due to a few very complex modules.

The main result of the empirical evaluation is the correlation of these measures
with maintenance change data. Modules with high information flow were well-know
by maintenance staff as being the source of many problems.

2.3.3.2 Object-Oriented Measures

Chidamber and Kemerer [1994] developed a set of class measures for OO designs.
They were evaluated theoretically against a set of measurement axioms and em-
pirically data was collected from two commercial systems. Some of the measures
presented are:

- **Weighted Methods per Class (WMC).** Consider a class $C_1$ with methods $M_1,
  M_2, \ldots, M_n$ defined in the class. Let $c_1, c_2, \ldots, c_n$ be the complexity of the
  methods. Then $WMC = \sum_{i=1}^{n} c_i$. The complexity of an individual method
  is not defined, in order to allow general application of this measure. If all
  method complexities are considered to be unity, then $WMC = n$, the number of
  methods.

- **Coupling between Objects (CBO).** CBO for a class is a count of the number of
  other classes to which it is coupled. CBO relates to the notion that an object
  is coupled to another if one of them acts on the other, i.e. a method of one
  object use methods or instance variables of another.

- **Response for a class (RFC).** $RFC = |RS|$ where $RS$ is the response set for the
  class. The response set for a class can be expressed as $RS = \{M\} \cup_{all-i}\{R_i\}$
  where $\{R_i\}$ is the set of methods called by method $i$ and $\{M\}$ is the set of all
  methods in the class. The response set of a class is a set of methods that can
  be potentially executed in response to a message received by an object of that
  class.

- **Lack of Cohesion in Methods (LCM).** Consider a class $C_1$ with $n$ methods $M_1,$
  $M_2, \ldots, M_n$. Let $\{I_i\}$ be the set of instance variables used by method $M_i$.
  There are $n$ such sets $\{I_1\}, \ldots, \{I_n\}$. Let $P = \{(I_i, I_j) | I_i \cap I_j = \emptyset\}$ and $Q =$
\{(I_i, I_j) | I_i \cap I_j \neq \emptyset\} \). Then, \( LCM = |P| - |Q| \) if \( |P| > |Q| \), \( LCM = 0 \) otherwise. \( LCM \) is the number of nonempty intersections. This measure uses the notion of "similarity" (i.e. number of shared instance variables) of methods to characterise how closely the local methods of a class are related to the instance variables.

The presented empirical data illustrate the potential use of these measures on real applications, but it is not used to analyse the degree to which these measures are related with attributes of interest such as productivity and development effort.

Lorenz and Kidd [1994] proposed a group of "design measures" to capture the static attributes of software design. The measures are categorised into class size measures, class inheritance measures, and class internals measures. We only present those that can be applied to UML class diagrams—see Table 2.2. To our knowledge no work related to the theoretical validation of these measures has been published. Practical recommendations were provided after applying these measures to 5 real projects. In order to collect these measures a tool was developed for code written in Smalltalk and C++.

### 2.3.3.3 Interaction-based Measures

Briand et al. [1999] introduce and compare various design measures in the context of systems developed with Ada. Specifically, they define a set of measures for cohesion and coupling, which are theoretically validated against a set of mathematical properties. Our research method has many aspects in common with the procedure followed in this work, which makes it particularly relevant.

The following definitions are presented to characterise the terminology used to define the measures:

- **Library Module Hierarchy (LMH).** A LMH is a hierarchy where nodes are modules and subroutines, arcs between nodes are IS-COMPONENT-OF relationships, and there is exactly one top-level node, which is a module.

- **High-level Design.** The high-level design of a software system is a collection of module and subroutine interfaces related to each other by means of USES and IS-COMPONENT-OF relationships.
<table>
<thead>
<tr>
<th>Category</th>
<th>Measure</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class size</td>
<td>PIM</td>
<td>The Number of Public Instance Methods in a class is the total number of public instance methods in a class. Public methods are those that are available as services to other classes. The number of public instance methods in a class is a measure of the amount of responsibility in the class.</td>
</tr>
<tr>
<td></td>
<td>NIM</td>
<td>The Number of Instance Methods in a class counts all the public, protected, and private methods defined for the class instances. The number of methods in a class relates to the amount of collaboration being used.</td>
</tr>
<tr>
<td></td>
<td>NIV</td>
<td>The Number of Instance Variables in a class is the total number of instance variables in a class. Instance variables include private and protected variables available to the instances.</td>
</tr>
<tr>
<td></td>
<td>NCM</td>
<td>The Number of Class Methods in a class is the total number of class methods in a class. A class method is a method that is global to its instances.</td>
</tr>
<tr>
<td></td>
<td>NVV</td>
<td>The Number of Class Variables in a class is the total number of class variables in a class.</td>
</tr>
<tr>
<td>Class Inheritance</td>
<td>NMO</td>
<td>The Number of Methods Overridden is the total number of methods overridden by a subclass. The measure looks at the quality of the classes' use of inheritance. It examines superclass-subclass inheritance relationships.</td>
</tr>
<tr>
<td></td>
<td>NMI</td>
<td>The Number of Methods Inherited is the total number of method inherited by a subclass. This measure also looks at the quality of the classes' use of inheritance.</td>
</tr>
<tr>
<td></td>
<td>NMA</td>
<td>The Number of Methods Added is the total number of methods defined in a subclass.</td>
</tr>
<tr>
<td></td>
<td>SIX</td>
<td>The Specialisation Index for each class is defined as:</td>
</tr>
</tbody>
</table>
|                         |         | \[
|                         |         | \[
|                         |         | \[
|                         |         | \[
|                         |         | \[
| Class Internals         | APM     | The Average Parameters per Method is defined as:                                                                                         |
|                         |         | \[
|                         |         | \[
|                         |         | \[

Table 2.2: OO Design Measures proposed by Lorenz and Kidd
2.3. SOFTWARE MEASUREMENT

- **Data Declaration-Data Declaration (DD) Interaction.** A module data declaration A interacts with another data declaration B if a change in A's declaration or use may cause the need for a change in B's declaration or use.

- **Data Declaration-Subroutine (DS) Interaction.** A module data declaration D interacts with a subroutine S if D interacts with at least one of S data declarations.

- **Cohesive Interactions.** The set of cohesive interactions in a module \( m \), denoted by \( CI(m) \), is the union of the set of DS-interactions and the set of DD-interactions involving exclusively data declarations and subroutines within \( m \), with the exception of those DD-interactions between a data declaration and a subroutine formal parameter.

- The maximal set of cohesive interactions in module \( m \) is denoted by \( MI(m) \). It is obtained by linking every data declaration \( D_i \) of module \( m \) to every other data declaration and subroutine of \( m \) with which \( D_i \) can interact.

Based on the above definitions, a measure is introduced to capture cohesion in their context: the **Ratio of Cohesive Interactions (RCI)**, which is defined for a software part \( sp^3 \) as:

\[
RCI(sp) = \frac{|SDD(sp)| + |CI(sp)|}{|SDD(sp)| + |MI(sp)| + |SSR(sp)|}
\]

where

- \( SSR(sp) \) denotes the set of subroutines belonging to modules of \( sp \) that do not contain any data declaration, and

- \( SDD(sp) \) denotes the set of modules of \( sp \) that only contains a single data declaration.

A threshold may be defined, based on this measure, to be used as a support for deciding whether a set of data and sub-routines should be kept in one single module or divided into two or more modules.

---

3 A software part denotes either a module or a LMH.
Since cohesion is not the only attribute relevant to software design, measures to capture coupling are also proposed. Software parts with low interaction coupling should be easier to analyse, understand and test separately. Import Coupling, $IC(sp)$, given a software part $sp$, is defined as the number of DD-interactions between data declarations external to $sp$ and the data declarations within $sp$. Based on this generic definition, they derive two related measures:

- Direct Import Coupling, $DIC(sp)$, only takes into account direct interactions, whereas
- Transitive Import Coupling, $TIC(sp)$, only takes into account transitive interactions,

with $IC(sp) = DIC(sp) + TIC(sp)$.

Their empirical evaluation, using data from systems of a NASA flight centre, shows that these measures are related to and can be used as indicators of fault-proneness in their context.

It should be noted that other measures were defined but are not reported here since they are not relevant to our research.

### 2.4 State of the art of software measurement for DIS

In this subsection we present existing proposals that address the need of measures for distributed software.

#### 2.4.1 Shatz [1988]

This author proposed a measure to quantify communication complexity in Ada programs. He suggested that the total complexity of a distributed Ada program is the weighted sum of two components:

- *local complexity*, which reflects the complexity of an individual task, disregarding its interactions with other tasks; and
- *communication complexity*, which reflects the complexity of interactions among the tasks.
Distributed Ada programs are represented using Petri nets, and communication complexity is defined in terms of the number of rendezvous that can be executed concurrently at a point in time.

The total complexity $TC$ of a distributed program is proposed as:

$$TC = X \times \sum_{i=1}^{T} (LC_i) + Y \times CC$$  \hspace{1cm} (2.1)$$

where $T$ is the number of tasks, $LC_i$ is the local complexity of the task $i$, $CC$ is the communication complexity of the tasks, and $X$ and $Y$ are weight values that would be used to adjust for the fact that the two types of complexities may not have the same impact in deciding the overall complexity.

However, the research only involves communication complexity, since it is stated that local complexity can be evaluated using existing measures of non-distributed programs complexity, such as lines of code, for the individual tasks. Consequently, the author considers program patterns of communication to capture the communication complexity and the measurement of concurrently active rendezvous as a useful component in deriving a communication complexity measure.

A Petri-net graph model is used to represent a distributed program. From this model a reachability graph can be generated. The maximum number of concurrently active rendezvous that can occur along a given program path is determined by tracing the program paths in the reachability graph and applying a token count function at each step. In the process of following a reachability graph path, the token count values can be used to plot a rendezvous graph that shows the sequence of rendezvous events.

Rendezvous graphs may be utilised to compute a numerical complexity score $S$, and once the scores values for each path are computed, an expected score for the program $PS$ can be obtained as follows:

$$PS = \sum_{i=1}^{P} (S_i \times Prob_i)$$  \hspace{1cm} (2.2)$$

where $P$ is the number of paths and $Prob_i$ is the probability of path $i$ being affected by future maintenance. The idea is that if a very complex path is unlikely to be
changed, the path's impact on the program's effective complexity should be scaled appropriately.

Although these ideas are not integrated into a concrete measure and the analysis is limited to Ada rendezvous, this is the first discussion about software measurement for distributed systems in the literature.

2.4.2 Cheng [1993]

This paper proposes some complexity measures for distributed programs. The measures are defined based on graph representations of control flow and primary program dependences in distributed programs: the non-deterministic parallel control-flow net (CFN) and the process dependence net (PDN).

The author states that complexity measures of non-distributed programs—defined based on traditional deterministic and sequential control-flow graphs—can be redefined or extended for distributed programs based on the CFN. For example, they contend that the cyclomatic complexity measure [McCabe, 1976], which is defined as the number of linearly independent paths through a control-flow graph, can be redefined for distributed programs as the number of linearly independent paths through its CFN.

Based on the PDN representation of a distributed program, they define some complexity measures in terms of program dependences and dependence types; control dependence, information dependence, selection dependence, synchronisation dependence and communication dependence. These measures include:

- For each type of dependence, the number of direct dependences.
- The number of all direct dependences.
- For each type of dependence, the maximal number of statements that a statement is directly dependent on.
- The maximal number of statements that a statement is directly dependent on.

Similar measures are also defined in terms of indirect dependences, to capture the impact of chains of primary dependences.
The paper presents two more graph-based representations—the \textit{non-deterministic parallel definition-use net} and the \textit{process influence net}—for representing information flows and program influences of distributed programs. Although it is suggested that complexity measures in terms of information-flow and program influences can be defined analogously, no details are given.

The measures are independent of any programming language since they are derived from a generic graph representation of a distributed program. However, no empirical validation is provided.

\textbf{2.4.3 Tsaur and Horng [1998]}

This research also discusses the complexity of distributed programs. Their focus is on the communication aspect of distributed programs. Based on the \textit{smallest event communication group} (SECG) concept, they propose a measure to quantify the software complexity for distributed systems.

They define total software complexity $TSC$ for a distributed program as follows:

$$TSC = X \times \sum_{i=1}^{n} |LC_i| + Y \times \sum_{i=1}^{m} |SECG_i|$$

where $n$ is the number of processes, $m$ is the number of SECG, $|LC_i|$ is the local complexity of process $i$, $X$ and $Y$ are weighted values, and $|SECG_i|$ is the number of events of a $SECG_i$.

They describe informally the execution of a distributed program as a distributed computation by a collection of processes. These processes perform operations independently and concurrently, and cooperate with each other by communication. The activity of each sequential process is modelled as executing a sequence of events. An event is called an \textit{internal} event if it is internal to a process, and it is called an \textit{external} event if it involves a communication with another process—send/receive.

A sequence of events in process $p_i$, denoted as $h_i = e_i^1 e_i^2 e_i^3 \ldots e_i^n$, constitutes its local history and a global history of a computation is the union of the local histories of all processes containing all of their events: $H = h_1 \cup h_2 \cup h_3 \ldots \cup h_n$, where

- $e_i^k$ denotes the $k$th event for process $p_i$, and

---

33
CHAPTER 2. BACKGROUND AND RELATED WORK

- $E_i$ denotes the event set for process $p_i$.

This view of a distributed computation as an event occurrence, together with the causality relation, leads them to a definition of program execution as a partial ordering of an event occurrence. Information may flow from one event to another, either because the two events are of the same process or because they are of different processes. These ideas are formalised defining a binary relation ‘→’ (happen before) over the events of the program:

- if $e_i^l, e_i^s \in$ and $l < s$, then $e_i^l \rightarrow e_i^s$;

- if $i \neq j$ and $e_i^s$ is a send event and $e_j^r$ is the corresponding receive event, then $e_i^s \rightarrow e_j^r$;

- if $e_i^s \rightarrow e_k^s$ and $e_k^s \rightarrow e_j^r$, then $e_i^s \rightarrow e_j^r$.

Certain events of the global history may be causally unrelated, since it is possible that for two distinct events $e$ and $e'$, neither $e \rightarrow e'$ nor $e' \rightarrow e$. Such events are called concurrent and are written $e \parallel e'$.

The notion of a distributed-computation cut is introduced as a subset $C$ of the global history $H$ that contains an initial prefix of each of the local histories. Such a cut $C$ can be specified through the tuple of natural numbers $(c_1, c_2, \ldots, c_n)$ corresponding to the index of the last event included for each process. A cut is consistent if for all events $e$ and $e'$, $(e \in C) \land (e' \rightarrow e) \Rightarrow e' \in C$.

They consider this view of a distributed computation at the event level as fundamental to capture its behaviour. However, it is often required to create multiple viewpoints on a program, and to be able to view the program at different levels of abstraction. Then, they also define an abstraction view of process communication, the event communication group (ECG), where a send event and its corresponding receive event are a primary process communication unit, as a nonempty set of events that have occurred between any two consistent cuts $C_1 = (c_1, c_2, \ldots, c_n)$ and $C_2 = (c'_1, c'_2, \ldots, c'_n)$, where $C_1 \neq C_2$, and $\forall k \in \{1, 2, \ldots, n\}, c_k \leq c'_k$.

To keep the size of the ECG manageable, the SECG is defined as an ECG that cannot be further divided. A SECG possesses computational independence since
no message passes between groups and every message sent must be received in the same group. In addition, it can be considered as an isolated communication unit with a well-defined interface, which is one of their criteria for modularisation.

In this paper they outline an algorithm to detect SECGs as they occur during the distributed computations. The measure is evaluated experimentally with an example.

2.4.4 Morasca [1999]

This paper proposes a set of measures for capturing a number of internal attributes, i.e. size, length, complexity, and coupling, of concurrent software specifications.

The basic constructs (i.e. sequence, decision point, parallelism and synchronisation) used to specify concurrent software are represented by Petri nets [Murata, 1989]. The static part of a Petri net, representing the structure of the modelled system, is a bipartite directed graph called a net. The nodes of a net are divided into places and transitions. Thus, a net $N$ is formally defined as $N = (P, T, F)$ where $P$, the set of places, $T$, the set of transitions, and $F$, the set of arcs, are such that

\[
\begin{align*}
\Box P \cap T &= \emptyset \\
\Box F &\subseteq (P \times T) \cup (T \times P)
\end{align*}
\]

Therefore, it is assumed that a Petri net $N$ is used to specify a concurrent software system, which has a nonempty set of places $\text{Pre}(N)$ with no incoming arcs and a nonempty set of places $\text{Post}(N)$ with no outgoing arcs. These two sets of places represent the start and end states of the processes of the concurrent software system.

Some of the defined measures are presented in Table 2.3. These measures are theoretically validated against a collection of sets of properties that have been proposed in the literature, but empirical validation is not addressed.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Measure</th>
</tr>
</thead>
</table>
| Size       | **Number of transitions:** $|T|$  
            | **Number of places:** $|P|$                                             |
| Length     | **MaxMinDistance:** the maximum number of arcs of the shortest paths from places in $Pre(N)$ to places in $Post(N)$. |
| Complexity | **Number of base paths:** $|F| - |T| - |P| + |Post(N)|$  
            | **Concurrency complexity:** $|F| - 2|T|$                                             |
| Coupling   | **Number of outbound arcs:** given a subnet $SN$ of a Petri net $N$, a measure of outbound coupling may be defined as the number of arcs whose source node belongs to $SN$, and whose destination node does not belong to $SN$.  
            | **Number of inbound arcs:** given a subnet $SN$ of a Petri net $N$, a measure of inbound coupling may be defined as the number of arcs whose destination node belongs to $SN$, and whose source node does not belong to $SN$. |

Table 2.3: Measures proposed by Morasca (1999)

### 2.4.5 Discussion

Our first observation regarding the state-of-the-art of distributed software measurement is that almost all these measures are derived from the code of a distributed program. Only one proposal [Morasca, 1999] addresses the measurement of early artefacts ( specifications ), although the measurement of design artefacts in particular, has not been attempted yet.

A second observation is that complexity, particularly communication complexity, is the attribute mostly targeted. This gives a strong indication that connectivity of distributed components is a key aspect to be considered when measuring distributed software artefacts. Furthermore, one of the proposals also recognises the importance of addressing coupling.

On further observation, we found that most of these measures are not defined formally nor evaluated theoretically. In some cases, empirical evaluation is not addressed [Shatz, 1988; Morasca, 1999]. Moreover, none of these proposals attempt to complement theoretical and empirical evaluation.

Finally, from the practical perspective, we noticed that although most measures

are independent of the implementation technology and objective, their capability of being collected automatically by a tool is not demonstrated.

Table 2.4 summarises the proposals for distributed software presented in this section. The keys aspects compared were:

- the attribute characterised and the artefact measured;
- the presence or absence of a formal definition or theoretical evaluation;
- the presence or absence of an empirical evaluation;
- the dependence or independence with regards to the implementation technology; and
- the presence or absence of an automatic collection tool.

2.5 Summary

In this chapter we have reviewed relevant work in software measurement, and we have examined existing measure proposals related to our work. We have also discussed the characteristics of DIS and highlighted some of the differences and extra issues that are found in this particular scenario. The research method followed in this dissertation is presented in the next chapter.
Chapter 3

Methodology

3.1 Introduction

A large number of measures have appeared in the literature to capture software product attributes in a quantitative way. However, few measures have successfully survived the definition phase and are actually used in research and/or practice. This is due to a number of problems related to the development of many measures, some of which are summarized by Briand et al. [2002]:

- Measures are not always defined in the context of some explicit and well-defined measurement goal derived from an objective of research/industrial interest they help reach.

- Even if the goal is made explicit, the experimental hypotheses are often not made explicit.

- Measurement definitions do not always take into account the environment or context in which they will be applied.

- A reasonable theoretical validation of the measure is often not possible because the attribute that a measure aims to quantify is often not well defined.

- A large number of measures have never been subject to empirical evaluation.

This does not imply that progress cannot be achieved in the software measurement field. A disciplined and structured approach to the development of measures
would allow practitioners and researchers to minimise—even avoid—these problems. Therefore, the objective of this chapter is to present the research methodology followed in this dissertation to develop software measures systematically.

3.2 Description of the Research Method

There is a considerably ample literature with guidelines on how to develop software measures. However, there is no clear consensus of how to do it, nor established terminology [Ruiz et al., 2003]. Although we have used the template proposed by Shepperd and Ince [1993] to derive our research method, it should be noted that the method also includes suggestions proposed by other sources—notably from Briand et al. [2002].

Figure 3.1 shows the high-level structure of the method. It should be noted that the method is iterative, since it may be necessary to return to previous stages when additional insight is achieved. The following sections describe in detail the different phases of the method.

3.2.1 Informal Model

The first stage is to create an informal model of the software entities (and attributes) to be measured. Experience, intuition and critical analysis play an important role here, to determine specific goals perceived to be important towards the problem domain.

This preliminary model, at this stage, should consider key (internal and external) attributes that may be relevant within the problem domain. Elementary relationships between the considered attributes must also be analysed.

It is important to note that there is not always a one-to-one mapping between a model and a goal. One goal may be addressed by several models, and one model may be useful for a multiplicity of goals.
Figure 3.1: Method for developing software measures
CHAPTER 3. METHODOLOGY

3.2.2 Formal Model

In order to progress with the development of measures, the collection of intuitions and insights of the informal model must be consolidated into a more formal model.

The first step of this stage is to establish explicit measurement goals. Then, a crucial task is to map the modelled relationships between the internal attributes and external attributes to experimental hypotheses. Finally, the abstraction of the software entity and attributes should be as formal as possible, since formal abstractions can be used to describe the model unambiguously and to determine whether the attributes can be measured directly and/or objectively.

Any assumptions that are made and the rationale for introducing parameters if appropriate should be documented.

3.2.2.1 Establishing Measurement Goals

The definition of measurement goals should be derived from the intuitive model(s) and should be in line with the research objectives. The GQM technique [Basili and Rombach, 1988] is commonly used as a template to define measurement goals explicitly along five dimensions:

- an object of study that helps to determine the entities to be represented by abstractions, such as a product or a process;

- a purpose that shows the potential use of the measures to be defined, such as understanding or evaluation;

- a quality focus that assists the choice of dependent (or external) attributes used in the hypotheses, such as maintainability or reliability;

- a viewpoint that identifies who is affected by the results, such as the software engineer or manager;

- a description of the environment to give proper context and scope to any results, such as a financial organisation or a manufacturing company.

For example, Plattini et al. [2001b] define their measurement goal as:
Analyse entity-relationship diagram measures, for the purpose of evaluation, with respect to maintainability sub-attributes, from the point of view of database/software designers, in the context of software development departments/companies.

### 3.2.2.2 Setting Experimental Hypotheses

Hypotheses on software attributes should be consistent with the measurement goals. An experimental hypothesis establishes an explicit link between the internal and the external attribute(s) of interest. That is, a relationship between the object of study and the quality focus—for example, the size of a software class is related to its fault-proneness. Hypotheses are also originated from the intuitive model, thus avoiding a random search for statistical significance.

Several different hypotheses may be proposed at this point. However, it should be noted that the hypotheses:

- are not universal, since a priori they are only relevant to the studied environment;
- are not unique in the studied environment, since they capture our beliefs and other researchers may posit other hypotheses;
- are not equally important, since the empirical evaluation determines whether they are (statistically) valid or not, and with what degree of confidence.

### 3.2.2.3 Identification of Entity Abstractions

An abstraction is a representation of an entity. Some examples of software (product) abstractions are UML class diagrams—which represent a set of classes and their relationships—and flowcharts—which represent the steps of an algorithm and their flows of control. The object of study may be mapped into one or more abstractions to capture the necessary information to quantify its attributes. Abstractions are commonly used in software engineering to capture some attributes of entities while abstracting other attributes.
The hypotheses defined in the previous step guide the selection of abstractions, since the represented software entities and their attributes are involved in those hypotheses. Some of the available abstractions in the literature can be reused or new representations may be created.

3.2.3 Definition of Measures and Theoretical Validation

A measure quantifies an intuitive concept such as size or cohesion, and may be loosely defined as the assignment of a number to an entity to characterise an attribute. For example, widely-known traditional measures that characterise software size include lines-of-code and function-points.

In this stage, the formal framework proposed by Poels and Dedene [2000] is applied to define measures rigorously. This axiomatic approach has been used by other researchers [Genero et al., 2002; Abrahao et al., 2003] and offers the following advantages:

- It guarantees the theoretical validity of the defined measures (according to the theory of measurement), while hiding the complexity of the underlying theoretical foundations.

- It produces ratio scale measures.

- It provides a constructive and systematic framework to define valid measures that is applicable to different types of software artefacts.

This formal approach to software measurement constructs a metric space \((M, \delta)\) for the set of measurement abstractions \(M\) that are used as software product representations to characterise the (internal) attribute of interest. A metric space is a tuple composed of a set and an associated function that defines the distance between the elements of the set. That is, if \(\delta : X \times X \rightarrow \mathbb{R}\) is a distance function—or metric—as defined below, then \((X, \delta)\) is a metric space.

After the construction of the metric space \((M, \delta)\), the attribute of interest is represented as the (conceptual) distance or dissimilarity from the measurement abstraction \(m\) of the software entity under consideration to some well-chosen ref-
3.2. DESCRIPTION OF THE RESEARCH METHOD

cercence abstraction \( r \). Finally, the numeric value returned by \( \delta(m, r) \) quantifies the attribute.

In Mathematics, distance functions (or measures of distance) are also called metrics. The function \( \delta : X \times X \to \mathbb{R} \) is a metric if it satisfies the following axioms:

- **non-negativity**: \( \forall x_1, x_2 \in X : \delta(x_1, x_2) \geq 0 \)

- **identity**: \( \forall x_1, x_2 \in X : \delta(x_1, x_2) = 0 \iff x_1 = x_2 \)

- **symmetry**: \( \forall x_1, x_2 \in X : \delta(x_1, x_2) = \delta(x_2, x_1) \)

- **triangle inequality**: \( \forall x_1, x_2, x_3 \in X : \delta(x_1, x_2) \leq \delta(x_1, x_3) + \delta(x_2, x_3) \)

If \( \delta : X \times X \to \mathbb{R} \) is a metric, then the value \( \delta(x_1, x_2) \) quantifies the distance between \( x_1 \) and \( x_2 \). For example, the metric of Euclid measures the distance between two points in the plane (i.e. \( d((x_1, y_1), (x_2, y_2)) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \)); with the binary metric there are no degrees of “closeness”: two points are either identical or they are different, i.e. \( d(x, y) = 0 \) if \( x = y \), otherwise \( d(x, y) = 1 \).

Although the term *metric* is often considered as synonym of the term *measure* in the Software Engineering literature (e.g. [Pressman, 2001], [Sommerville, 2001]), in this dissertation we consider it important to distinguish between the two terms due to this theoretical approach.

This framework ensures that the metric defined for the measurement of an attribute is a theoretically valid measure in the sense of measurement theory, and that it measures what it is purported to measure. Software attributes are modelled with a segmentally additive proximity structure and the measures are defined using a metric with additive segments [Suppes et al., 1989].

The following constructive procedure can be used to define software product measures for the set of software entities (products) \( E \) and for the internal attribute \( a \) of the software entities in \( E \) [Poels and Dedene, 2000]:

1. **Choose a set of software artefacts** \( M \) **that can be used as measurement abstractions to emphasise** \( a \), **and define a mapping** \( \alpha : E \to M \). The function \( \alpha \) is total, but is neither surjective nor injective. That is, respectively, each relevant abstraction of a software entity in \( E \) is contained in \( M \), but not
all elements of $M$ are measurement abstractions of software entities in $E$, and different software entities in $E$ may be mapped to the same element of $M$. It should be noted that for some attributes there might exist different types of measurement abstractions that model the same attribute. In these cases, the attribute actually consists of a number of sub-attributes, one for each type of measurement abstraction. The application of this approach to each sub-attribute results in a different measure for each sub-attribute. The use of different abstractions implies that different notions of the same attribute are measured, unless the abstractions are equivalent\(^1\). For example, if control-flow graphs are the software entities, we can use the set of nodes of the graph as the measurement abstractions to characterise the attribute size.

2. **Define a set $T$ of elementary transformations on $M$ that is constructively and inverse constructively complete.** In this step distances between the elements of $M$ are modelled. This will allow the distinction of the software entities in $E$ in terms of $a$. Distances between the elements of $M$ are defined in terms of elementary transformations. These represent atomic changes that transform one element of $M$ into another element of $M$. Intuitively, the number of atomic changes required to transform one element into another defines the distance (or dissimilarity) between those elements. Each elementary transformation of an element of $M$ is formalised as a function $\theta_i : M \rightarrow M$ that transforms the elements of $M$ according to some prescribed rules. The set $T$ of elementary transformations is required to be constructively and inverse constructively complete\(^2\). In this way, any element of $M$ can be transformed into any other element of $M$. For example, if the set of nodes of a control-graph is used as measurement abstraction, the transformations ‘add node’ and ‘remove

---

\(^1\)It is important to distinguish between entity abstractions and measurement abstractions. An entity abstraction is a representation of the software entity to be measured, where one or more attributes can be captured. A measurement abstraction is a simplification of an entity abstraction that is meant to emphasise a specific attribute being measured.

\(^2\)This means that:

- there is an initial abstraction $m_0$ in $M$;
- each element of $M$ can be built by applying a finite sequence of elementary transformation of $T$ to $m_0$; and
- for each elementary transformation $\theta$ included in $T$, the inverse of $\theta$ is also contained in $T$.  

46
node\' are sufficient to form a constructively and inverse constructively complete set of elementary transformations \( T \) on this measurement abstraction.

3. **Define a metric** \( \delta : M \times M \to \mathbb{R} \) **such that** \( (M, \delta) \) **is a metric space.** A metric is proposed in this step to quantify the distances in \( M \) that were defined using \( T \). Basically, the function \( \delta \) returns for each pair of elements in \( M \) the minimum number of elementary transformations that is needed to take one element to the other. Although this number may be multiplied by any positive real number \( k \), generally it is assumed that \( k = 1 \).

4. **Choose a reference abstraction** \( \tau \in M \) **for which it holds that for all** \( e \in E \) **with** \( \alpha(e) = \tau \), \( e \) **has the theoretical lowest value of** \( a \). This is a crucial step since it reflects the understanding of the attribute being measured. The selection of a reference model for an attribute is subjective and can be based on intuitive or empirical arguments. If appropriately chosen, then it can be intuitively argued that, for a software entity \( e \) the value of \( a \) is low when \( \alpha(e) \) is "close to" \( \tau \) and the value of \( a \) is high when \( \alpha(e) \) is "far from" \( \tau \). That is, it can be said that the distance from \( \alpha(e) \) to \( \tau \) characterises the attribute \( a \) of the entity \( e \). It is common that the reference abstraction \( \tau \) is exactly the same as the initial abstraction \( m_0 \). For example, the empty set of nodes could be used as a reference abstraction for the graph size. Such a hypothetical graph without nodes would have the lowest value (zero) for size as considered here.

5. **Define a function** \( \mu : E \to \mathbb{R} \) **such that for all** \( e \in E \), \( \mu(e) = \delta(\alpha(e), \tau) \). The desired measurement value of the attribute \( a \) of an entity \( e \) is the value returned by the metric \( \delta \) given a pair of abstractions in \( M \) consisting of the measurement abstraction of \( e \) and the reference abstraction. For example, if \( \mu \) is a measure of program size (in terms of its control-flow graph), then \( \mu \) returns for a program represented by the control-flow graph \( g \) of Figure 3.2, the count of elementary transformations in the shortest sequence from set of nodes of graph \( g \) to the empty set. That is \( \mu(p) = 4 \).
3.2.4 Empirical Evaluation

In this stage the original experimental hypotheses must be refined based on specific measures for both the internal and the external attributes. Measures for the internal attributes have already been defined in the previous stage, thus it is necessary now to define measures for the external attributes. Once the hypotheses have been refined in terms of actual measures, it is possible to state more precise relationships between the internal and external measures, consistent with the initial hypotheses. Based on the measures, data collection must be planned and performed to evaluate the hypotheses. Afterwards, the data collected may be used to test the hypotheses and to determine to what extent the independent measures are of practical interest.

The procedure to follow largely depends on the measurement goals, and different techniques may be used. In this dissertation we use correlation and regression analysis since they are robust techniques suitable for exploratory research, and they are commonly used in Software Engineering research [Briand et al., 1995]. Correlation analysis enables us to assess the strength of the relationships between variables, while regression analysis provides the basis for forecasting the values of a variable from the values of one or more variables.

Equations are used in Mathematics to express the relationships between variables. These equations express the deterministic relationships among the variables of interest. For example, the equation \( a = s^2 \) describes the relationship between \( s \) (the length of the side of a square) and \( a \) (the area of the square). By substituting numerical values for the variables on the right-hand side of these equations, we
can determine the exact value of the quantities on the left-hand side.

In Software Engineering and other disciplines such as the social sciences, deterministic or exact relationships are not generally observed among variables, but rather stochastic or statistical relationships prevail. That is, certain average relationships may be observed among variables, but these average relationships do not provide a basis for perfect prediction. For example, if the number of function points of an application is known, an exact prediction of the final size of the application cannot be made. However, on average, it is possible to statistically measure how size varies with differences in function points. We can also determine to what extent actual figures vary from these average relationships. Furthermore, on the basis of these relationships, the values of the variables of interest may be estimated closely enough to be able to make managerial or engineering decisions. The techniques of correlation and regression analysis are important statistical tools to study the relationships between variables of interest [Hair et al., 1995].

The basic objective of correlation analysis is to describe the degree to which one variable is associated (or related) to another, that is how well they are correlated. Basically there are three ways in which two variables could be related:

- they could be positively correlated,
- they could be not correlated at all, or
- they could be negatively correlated.

The most commonly used correlation statistic is the Pearson coefficient \( r \). This coefficient can range from \(-1\) to \(+1\), with a value of \(0\) indicating no linear relationship. A positive value means that there is a positive relationship and a negative value means that there is a negative relationship. Values close to \(0\) denote weak relationships, while values close to \(\pm 1\) indicate strong relationships. The Pearson correlation coefficient \( r \) assumes that the distribution of variables is normal. If this assumption is not tenable, a special case of the Pearson coefficient may be used instead: the Spearman coefficient rho \( \rho \).

Correlation analysis describes how closely a set of data points cluster around a straight line, but it does not provide information about the equation that describes
the line best and summarises that set of data points. In other words, correlation is useful to assess the strength of variable relationships but it tells us nothing about the explanatory (or predictive) power of variables.

The purpose of regression analysis is to provide estimates of a variable from the values of one or more other variables by estimating the parameters of the equation linking them. A (simple linear) regression model is the best-fitting straight line given a set of data. A regression line is described by equation 3.1, in which \( Y \) is the response variable and \( X \) is the predictor (or explanatory) variable. \( \beta_1 \) is the gradient of the line fitted to the data and \( \beta_0 \) is the intercept of that line. There is also a residual term, \( \varepsilon \), which is a random error that represents difference between the predicted value and the actual value. Since it is possible to describe a line knowing only the gradient and the intercept, the values of \( \beta_0 \) and \( \beta_1 \) may be used to describe a simple linear regression model.

\[
Y = \beta_0 + \beta_1 X + \varepsilon \tag{3.1}
\]

Simple regression seeks to predict the value of a response (or dependent) variable from a single explanatory (or independent) variable. Multiple regression models are a logical extension of simple regression where the value of a response variable is predicted from several explanatory variables. A similar equation can be derived to describe a multiple (linear) regression model (see equation 3.2). \( Y \) is the response variable, \( X_i \) is the \( i \)th predictor variable, \( \beta_i \) is the coefficient of the \( i \)th predictor \( (X_i) \), and \( \varepsilon \) is the error term.

\[
Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n + \varepsilon \tag{3.2}
\]

Although this model is more complicated, the basic purpose is the same as simple regression (i.e. to find the linear combination of predictor variables that correlate maximally with the response variable). For example, if we consider a multiple regression model with only two predictor variables, the model describes the plane in the space that best fit the observed set of data points.

The method of least squares is commonly used to fit regression models. This method determines the model of best fit, which minimises the difference between
the data points and a model. The differences between the data points and the model are called residuals, and provide information to assess how well a particular model fits the data (the goodness-of-fit of the model). Three complementary statistics may be used to evaluate the goodness-of-fit of a model [Freund and Wilson, 1998]:

- $R$ is a gauge of how well the model fits the observed data,
- $R^2$ is the amount of variation in $Y$ that is explained by the model from the sample data, and
- Adjusted $R^2$ is the amount of variation in $Y$ that would be explained if the model would have been derived from the population.

However, in order to generalize the results outside the sample data (i.e. for the whole population), we must examine the residuals and check that the underlying assumptions of the regression model are not violated [Chatterjee et al., 2000]:

- the residuals are normally distributed,
- the residuals have a constant variance, and
- the residuals are independent of each other.

Finally, it is also necessary to check that the fitted model is not affected by multicollinearity—i.e. a strong correlation between two or more predictor variables.

### 3.2.5 Practical Applicability Analysis

In this stage we analyse the practical applicability of the proposed measures. Many authors suggest that software measures should be validated, but also that to be useful in real projects they should be characterised by practical properties [Grady and Caswell, 1987; Card and Glass, 1990; Jones, 1991].

Several issues have been identified that are of concern to practitioners [Henderson-Sellers, 1996]. First, software measures should be automatible, otherwise they will not be collected. Second, it is also suggested that software measures should be objective and language independent. If the measures are not language independent, they have a limited scope of use and comparison across different projects is not
possible. If measures are not objective, their accuracy is tied on the expertise of the user. Finally, measures should be part of the software engineering process and provide informative and timely feedback, otherwise they will not be used.

3.3 Summary

In this chapter we have outlined the research method followed to develop measures systematically. Key guidelines and principles from the literature have been taken into consideration to structure the methodology. The next chapter presents the informal and formal measurement models.
Chapter 4

Measurement Models

4.1 Introduction

This chapter presents the application of the two first stages of the research method. Section 4.2 describes the informal model based on our intuitive analysis of the problem domain. The collection of intuitions and insights of the informal model are consolidated into a formal model in section 4.3.

4.2 Informal Model

A DIS is defined informally here as a collection of (possibly remote) software components structured to cooperate with each other to achieve common execution aims, and supported by some middleware infrastructure. A component is an autonomous software unit that, once constructed, is capable of being executed as a stand-alone process independently of the rest of the system.

The overall quality of a DIS can be considered a combination of two basic constituents:

- \textit{intrinsic} quality, which reflects the quality of the individual system components, disregarding their cooperation with other components; and

- \textit{extrinsic} quality, which reflects the quality of the cooperation among the system components.
Intrinsic quality can be evaluated with existing traditional approaches for centralised information systems. Hence, in this dissertation, we focus in the extrinsic quality of components. The examples presented in Table 4.1 illustrate why it is important to re-analyse some common traditional beliefs and assumptions in the case of the distributed scenario.

There is no causal relation between these quality aspects and the complexity aspects of Shatz [1988]. The only relation, which is casual, is that in both cases the attributes (i.e. quality and complexity) are both considered from two different but complementary perspectives.

Our objects of study are design artefacts of a DIS. More specifically, we are interested in two different but complementary artefacts:

- a structural artefact that captures static coupling of system components and their clustering into sub-systems; and
- a behavioural artefact that captures dynamic coupling among system components and their collaboration in cooperative scenarios.

We believe that the extrinsic quality of components is directly affected by the coupling among components from both perspectives.

### 4.2.1 Structure

The structure of a system is an important artefact, since it is available early on and provides key aspects to consider such as components, dependencies and clusters. Taking a component as the fundamental unit, dependencies between components are the way by which one component may affect another component. Our intuition is that the quality attributes of a component are easier to control the more autonomous it is. This is consistent with the notion of coupling in the centralised case. If there is a possibility of dependencies, additional components must be taken into account to evaluate its extrinsic quality. Thus, the fewer dependencies that exist between a component and other components, the easier it is to control its quality.

System components are usually grouped into clusters to support functional or
Example 1: Let Y be a procedure-oriented centralised information system that includes two modules C and D, and let Y’ be a procedure-oriented DIS that includes two remote modules C’ and D’. Assume module C is only coupled to module D by one (local) procedure call, this procedure does not return any value and requires a single parameter, which is purely data. Module C’ is only coupled to module D’ by one (remote) procedure call, this procedure does not return any value and requires a single parameter, which is purely data. In the case of the system Y, we can typically conclude that the coupling is minimal and further analysis is not required. Conversely, in the case of system Y’ the same conclusion does not apply since a remote procedure call is not the only choice offered by most existing middleware. For example, an (available) remote procedure call from C’ to D’ introduces more coupling, in a distributed computing sense, than a (non-available) message-based communication. Since module C’ does not really need module D’ to be available at the time of the communication, the extrinsic (and the total) reliability of C’ is disrupted should D’ be busy or off-line at the time of the remote procedure call.

Example 2: Let X be an object-oriented centralised information system that includes two objects A and B, and let X’ be an object-oriented DIS that includes two remote objects A’ and B’. Assume object A is only coupled to object B by one (local) method invocation, this method does not return any value and requires a single parameter, which is purely data. Object A’ is only coupled to object B’ by one (remote) method invocation, this method does not return any value and requires a single parameter, which is purely data. In the case of the system X, we can typically conclude that the coupling is minimal and further analysis is not required. In contrast, in the case of system X’, the same conclusion does not apply. For example, a synchronous invocation of the method from A’ to B’ implies more (distributed) coupling that an asynchronous invocation. Since object A’ would be blocked even when it does not really need to wait for the execution of the remote method to complete, the extrinsic (and total) efficiency of A’ is compromised unnecessarily.

Table 4.1: Differences between the distributed case and the non-distributed case in the evaluation of the extrinsic quality of components
geographical related business processes. One of the main drivers of distribution is to allocate components that support some business process physically close to it to maximise the degree of local (administrative and processing) autonomy. The rationale behind this is to ensure that local users have greater control of the components they operate, and to reduce their exposure to problems outside their control. That is, it is desirable that the DIS follows the structure and layout of the enterprise as much as possible. Each cluster has some boundary, and the interactions with other clusters occur via this boundary. Dependencies that cross this boundary are more difficult to implement and to maintain than dependencies that do not cross the cluster boundary. Therefore, we argue that intra-cluster dependencies and inter-cluster dependencies must be considered separately since their impact of software quality attributes may be different.

As introduced here, dependencies are a consequence of requirements. A decision as to whether a dependency is processing or informational, for example, depends on the analysis of the business process, not on any aspect of the structure of the system [Wijegunaratne and Fernández, 1998]:

- *Processing Dependency*: Component A needs some processing to be performed by component B before it can proceed executing. If the operation involves B and several other components, and is to be performed in all or nothing fashion, the processing dependency is actually a transactional dependency.

- *Informational Dependency*: There is no processing by component B required by component A. As a consequence of some event, component A only needs to convey some information to component B.

We contend that the system structure must follow the analysis of the business processes by designing dependencies among the components that are appropriate for the requirements. Thus, demarcating cluster boundaries is a key issue when designing the structure of a DIS. Although this determination depends on the specific problem at hand, components should be clustered in such a way that the coupling due to inter-cluster dependencies—specially processing dependencies—is minimised. It is widely acknowledged that a system should be partitioned into
sub-systems that exhibit low coupling and high cohesion.

Enterprise DIS are often large. Thus, it may not be possible or practical to allocate the same amount of resources for the development or testing of every single cluster of components. Nonetheless, we argue that if the allocation of resources is decided based on the structural attributes of clusters of components, it is possible to evaluate the quality attributes of clusters more effectively. The higher the inter-cluster coupling, the higher the potential sensitivity of a cluster to attributes of other clusters, and hence the higher their reliance on other clusters. For a cluster to achieve maximum autonomy, inter-cluster coupling should be kept to a minimum to reduce the impact of other clusters on its extrinsic quality attributes.

4.2.2 Behaviour

The system structure is usually available early but it only provides a static view of a distributed system. Coupling is not only affected by static aspects, but it is influenced by dynamic aspects as well. The study of the structure of a DIS, which gives an early indication of the quality of a system, can be enriched with a later study of its behaviour, obtaining supplementary information to better estimate quality attributes of the system.

The analysis of its behaviour captures key dynamic aspects of a DIS, since it can take into account the different modes of interactions between distributed components. The different possible modes of interaction between two running components have been described previously [Fernandez and Wijegunaratne, 1998] and are summarised here:

- **Synchronous vs. asynchronous**: Synchronous communication between two components occurs when one component makes a request to the other, and blocks while waiting for the reply. In asynchronous communication, the calling component does not block, but continues executing while waiting for the reply.

- **Available vs. non-available**: Available communication requires both interacting components to be available at the time of communication. Non-available
communication can proceed even if the called component is unavailable at the time of communication.

- **Conversational vs. non-conversational**: In a conversational interaction, a session is established between the two components, during which typically both components exchange information; the connection remains open during the whole session. Non-conversational communication is instead reduced to a one-off exchange, after which the connection is broken. If there is a need for more communication, a new connection needs to be established.

- **Static vs. dynamic binding**: Binding refers to the time at which some parameters of the communication are decided. Static binding binds communication parameters at compilation time, while dynamic binding leaves aspects of the communication to be decided at the time of execution.

It is important to note that the different modes of communication presented above are not mutually exclusive and refer to aspects that, with the exception of static binding, are relevant at execution rather than compilation time.

In the centralised case, dependencies between two components are almost always implemented as interactions resolved at compilation time, where the calling component is assumed to be accessible and it is assumed to block when the interaction occurs. That is, it is the case of a synchronous, available and statically bound interaction. In the distributed case, dependencies can be implemented in different ways due to the variety of possible interaction modes afforded by the middleware, therefore this aspect should be re-analysed. Taking a component as the fundamental unit of a distributed system, interactions between components are the way by which one component may affect another component dynamically.

The following discussion argues how the different types of dynamic interaction may impact on quality attributes.

**Synchronous vs. Asynchronous**. The impact of non-blocking calls on the efficiency of remote methods has received research attention in the OO area [Kerry Falkner et al., 1999; Maassen et al., 1999]. These efforts intend to improve efficiency when remote invocations are decoupled by
4.2. INFORMAL MODEL

calling a remote method at an earlier section of the code than the one where the returned value is used. In between the invocation and the use of the value the calling object does not block, but it continues executing. In this way, the calling object is not being tied to the status of the remote object, resulting in lower coupling. In this case, however, the remote object must be available to respond to the call. Therefore, we must distinguish between the two situations: a) a connection is made to obtain a value to be used in the calling component; and b) a connection is made to change the state of the remote component or, transitively, of other remote components. The difference in efficiency is significant, because in the first case the calling component eventually will have to request the result value and block if this is not available, while in the second case the execution may eventually terminate without blocking. Hence, the second situation presents lower coupling than the first. Reliability also seems to be an attribute favoured by asynchronous interactions, since the caller can continue its execution without failing if the called component is busy or temporarily off-line. If the connection is of type b) above, maintainability is likely to be higher. Since the called component can be taken off line without affecting callers' execution, the capacity of the component to be analysed and changed is improved.

Available vs. Non-Available. The invocation of type b) presented above is a particular case of non-available communication. In this type of communication the called component is not required to be available, so at the calling side the interaction is independent of the status of the called component; simply, the parameters of the call have to be assembled and marshaled, and the call made. If the interaction is of the type available, however, the unavailability of the remote component will have a negative impact on the efficiency of the caller, since the caller will have to wait, or the communication will have to be broken off after a given period and re-established at a later time. If availability is not required, the system is fault tolerant to a remote component being down or off-line, and com-
ponents are more likely to be oblivious to remote component failures, so
the overall reliability is increased. Finally, if the called component can
be taken off line without affecting callers' execution, the capacity of the
component to be modified and tested increases, and that means better
maintainability.

**Conversational vs. Non-Conversational.** A non-conversational inter-
action requires the caller to assemble and marshal the parameters of
the call and establish the connection, which breaks off after the required
values, if any, are returned. If further communication is required, a com-
pletely new similar sequence is necessary. Setting up a conversational
interaction typically takes more time, since the communicating objects
need to establish a session and the necessary structures (such as buffers
and pointers) to exchange information. The difference in efficiency be-
tween these interactions is largely dependent of the performance of the
underlying protocol, the length (time-wise) of the interaction, the likeli-
hood of successive remote invocations, and the volume of the informa-
tion exchanged. Thus, it is not possible to determine a general trend
with regards to efficiency, and the answer will largely depend on the spe-
cific case being considered. With regards to reliability, it is clear that
the longer the communication stays alive, the higher the probability of
failure and, therefore, conversational communication is likely to present
more robustness problems than non-conversational communication.

**Early/Static vs. Late/Dynamic Bound.** Efficiency in the static case is
generally better than in the dynamic case. This is because dynamic bind-
ing requires an extra level of indirection when the call is resolved. For
example, the list of parameters must be searched for, determined and
built (out of a registry or directory) or the correct network node must be
found at execution time. However, in terms of maintainability the cou-
pling of the dynamic version is always lower, because a statically bound
call is more susceptible to changes being performed in the called com-
ponents. Reliability also seems to be favoured in the late/dynamic case.
For instance, if the parameters of the call are discovered at run time, the
caller is impervious to changes on the remote component interface, since
there is no need for a priori knowledge of call information. The same ap-
plies to late binding to a network node: the caller remains resistant to
changes in the location of the remote component.

The preceding discussion can be illustrated with an intuitive analysis of specific
instances of design attributes affecting quality attributes.

4.2.2.1 Efficiency Analysis

As discussed above, the impact of non-blocking calls on the performance of remote
object methods has received significant research attention. These efforts attempt to
measure the gain in performance when remote invocations are decoupled, either by
using extended communication protocols or by embedding threads into a standard
protocol result in calling the remote method at an earlier section of the code than
the one where the returned value is used. In between the invocation and the
use of the returned value, the calling component does not block, but it continues
executing thereby improving performance due to the concurrent execution of the
components

Let us consider the following definitions:

- \( t_{i,j} \) is the execution time between the asynchronous call to remote component
  \( j \) from component \( i \), and the use of the return value—if the call is synchronous
  then \( t_{i,j} = 0 \);

- \( et_i \) is the time that component \( i \) spends executing without blocking due to
  remote calls;

- \( rtc_{i,j} \) is the time spent to transmit the parameters over the communication
  infrastructure, when component \( i \) calls remote component \( j \);

- \( bt_i \) is the blocking time of component \( i \) while a remote call to component \( j \) is
  being satisfied and \( bt_i = rtc_{i,j} + rt_j \) —if component \( i \) has no remote calls then
  \( bt_i = 0 \);
rti is the response time of component i to satisfy a call and \( rti = eti + bti \).

For simplicity, we first consider a distributed component with a single, simple remote call. The blocking time of component 1 when making a synchronous remote call to component 2 is \( bt1 = rtc1,2 + rt2 = rtc1,2 + et2 \), since it is the accumulation of the time spent on the communication infrastructure and the time required to satisfy the call by the remote component. However, if the remote call is asynchronous, the value is:

\[
bt1 = \begin{cases} 
0 & \text{if } t_{1,2} \geq rtc_{1,2} + rt2 \\
rtc_{1,2} + rt2 - t_{1,2} & \text{if } t_{1,2} < rtc_{1,2} + rt2 
\end{cases}
\]

where \( rt2 = et2 + bt2 = et2 \), because the calling component continues executing concurrently while the call is being satisfied. Hence, in the best case—when the calling component uses the return information after the call has already finished—there is no need to block at all. Even in the worst case, the asynchronous instance is more efficient than the synchronous in terms of performance.

If calls are chained as shown in Figures 4.1(a) and 4.1(b), the blocking time for component 1 is \( bt1 = rtc1,2 + rt2 = rtc1,2 + et2 + bt2 \), in both cases, since the chains begin with synchronous calls. However, \( bt2 \) is different in each case. In the case of Figure 4.1(a), \( bt2 = rtc2,3 + rt3 \), while in the case of Figure 4.1(b):

\[
bt2 = \begin{cases} 
0 & \text{if } t_{2,3} \geq rtc_{2,3} + rt3 \\
rtc_{2,3} + rt3 - t_{2,3} & \text{if } t_{2,3} < rtc_{2,3} + rt3 
\end{cases}
\]

where \( rt3 = et3 + bt3 = et3 \). In the case of a chain of calls, the inclusion of asynchronous calls improves the performance of the initial caller. Furthermore, if the chain starts with an asynchronous call—see Figures 4.1(c) and 4.1(d)—the blocking time could be lower (even zero if \( t_{1,2} \leq rtc_{1,2} + rt2 \)) and therefore, the response time will be also lower than in the previous case.

In the case of Figure 4.1(c), the blocking time of component 1 is

\[
bt1 = \begin{cases} 
0 & \text{if } t_{1,2} \geq rtc_{1,2} + rt2 \\
rtc_{1,2} + rt2 - t_{1,2} & \text{if } t_{1,2} < rtc_{1,2} + rt2 
\end{cases}
\]

where:

62
Figure 4.1: Remote calls among distributed components (efficiency)
\( \text{rt}_2 = \text{et}_2 + \text{bt}_2 = \text{et}_2 + \text{rtec}_{2,3} + \text{rt}_3, \)

\( \text{rt}_3 = \text{et}_3 + \text{bt}_3 = \text{et}_3. \)

In the case of Figure 4.1(d), the blocking time of component 1 is

\[
\text{bt}_1 = \begin{cases} 
0 & \text{if } t_{1,2} \geq \text{rtec}_{1,2} + \text{rt}_2 \\
\text{rtec}_{1,2} + \text{rt}_2 - t_{1,2} & \text{if } t_{1,2} < \text{rtec}_{1,2} + \text{rt}_2
\end{cases}
\]

where:

\( \text{rt}_2 = \text{et}_2 + \text{bt}_2. \)

\( \text{bt}_2 = \begin{cases} 
0 & \text{if } t_{2,3} \geq \text{rtec}_{2,3} + \text{rt}_3 \\
\text{rtec}_{2,3} + \text{rt}_3 - t_{2,3} & \text{if } t_{2,3} < \text{rtec}_{2,3} + \text{rt}_3
\end{cases}
\)

\( \text{rt}_3 = \text{et}_3 + \text{bt}_3 = \text{et}_3. \)

The following ranking presents the four cases of chains discussed above, from the worst to the best:

\( \text{case of Figure 4.1(a)}. \)

\( \text{case of Figure 4.1(b)}. \)

\( \text{case of Figure 4.1(c)}. \)

\( \text{case of Figure 4.1(d)}. \)

This analysis can be extended to deal with the general case—Figure 4.1(e). The following formula calculates the blocking time of the first component of a linear chain of remote calls:

\[
\text{bt}_1 = \begin{cases} 
0 & \text{if } RC = \emptyset \\
\alpha_{i,j} \ast (\text{rtec}_{i,j} + \text{rt}_j - \gamma_{i,j} \ast t_{i,j}) & \text{otherwise}
\end{cases}
\]

where:

\( RC = \text{set of remote components called by component } t, \)

\( \gamma_{i,j} = \begin{cases} 
0 & \text{if } j \in SC \\
1 & \text{otherwise}
\end{cases} \)
\[ \alpha_{i,j} = \begin{cases} 
0 & \text{if } t_{i,j} \geq rtc_{i,j} + rt_j \\
1 & \text{otherwise} 
\end{cases} \]

- \( SC \) = set of remote components synchronously called by component \( i \).
- \( rt_j = \text{et}_j + \text{bt}_j \).

Although this formula covers a wide range of cases, it does not cover all the possible remote call chains. Nevertheless, the following formula calculates the blocking time of the first component of a network of remote calls—see for example, Figure 4.1(0):

\[ bt_i = \begin{cases} 
0 & \text{if } RC = \emptyset \\
\sum_{j \in RC} \alpha_{i,j} * (rtc_{i,j} + rt_j - \gamma_{i,j} * t_{i,j}) & \text{otherwise} 
\end{cases} \]

where:

- \( RC \) = set of remote components called by component \( i \),
- \( \gamma_{i,j} = \begin{cases} 
0 & \text{if } j \in SC \\
1 & \text{otherwise} 
\end{cases} \)
- \( \alpha_{i,j} = \begin{cases} 
0 & \text{if } t_{i,j} \geq rtc_{i,j} + rt_j \\
1 & \text{otherwise} 
\end{cases} \)
- \( SC \) = set of remote components synchronously called by component \( i \),
- \( rt_j = \text{et}_j + \text{bt}_j \).

### 4.2.2.2 Reliability Analysis

An invocation made to send a message or change the state of a remote component usually does not require a return value. If middleware is used to implement the call in this type of communication, the called component is not required to be available, only the parameters of the call have to be assembled and the call placed on the communication infrastructure. The middleware guarantees delivery by saving first to non-volatile storage and repeating the message as required, taking care of details such as acknowledgements. However, if the invocation—such as synchronous or
asynchronous RPC—is of the type available, unavailability of the remote component will have a detrimental effect since the caller will have to block waiting, or break off the communication to re-establish it later. Thus, in a non-available invocation the overall system is more fault-tolerant to a remote component being busy, down or off-line, and the overall probability of failure-free operation is increased.

Let us consider the simple case of a remote call between distributed components and the following definitions:

- $n_f_i$ is the probability of non-failure when no remote calls are made from component $i$;
- $dn_f_i$ is the probability of non-failure of component $i$ when remote calls can be made—if no remote calls are made then $dn_f_i = n_f_i$;
- $n_f_c_{i,j}$ is the probability of non-failure of the communication infrastructure between component $i$ and $j$.

Then, if component $i$ makes a remote call to component $j$, its probability of non-failure is $dn_f_i = n_f_i * n_f_c_{i,j} * dn_f_j = n_f_i * n_f_c_{i,j} * n_f_j$ in the available case, since it is the product of the probabilities of the three independent events; and $dn_f_i = n_f_i * n_f_c_{i,j}$ in the non-available case, since a failure of component $j$ does not affect component $i$ if the call was successfully placed in the communication infrastructure. As the value of $n_f_j$ is less than one, $dn_f_i$ is higher in the non-available case than in the available case.

If calls are chained as shown in Figures 4.2(a) and 4.2(b), the probability is $dn_f_1 = n_f_1 * n_f_c_{1,2} * dn_f_2$ since in both cases the chain begins with an available call. However, $dn_f_2$ is different in each case: $dn_f_2 = n_f_2 * n_f_c_{2,3} * dn_f_3 = n_f_2 * n_f_c_{2,3} * n_f_3$ in the case of Figure 4.2(a); and $dn_f_2 = n_f_2 * n_f_c_{2,3}$ in the case of Figure 4.2(b).

It can be seen that in the case of a call chain, the inclusion of non-available calls provides a higher non-failure probability to the initial caller. Furthermore, if the chain starts with non-available call—Figures 4.2(c) and 4.2(d)—the non-failure probability could be even higher than in the previous cases, since the rest of the chain is decoupled. Therefore, in the cases of Figures 4.2(c) and 4.2(d) the non-failure probability of component 1 is $dn_f_1 = n_f_1 * n_f_c_{1,2}$.
Figure 4.2: Remote calls among distributed components (reliability)
In other words, if the first call is non-available, \( dnf_i = nf_i \ast nc_{i, 2} \) regardless of the second call. The following scale presents the four cases of chains discussed above from worst to best:

- case of Figure 4.2(a),
- case of Figure 4.2(b), and
- cases of Figures 4.2(c) and 4.2(d).

This analysis can be extended to deal with the general case—Figure 4.2(e). The following formula calculates the non-failure probability of the first component of a linear chain of remote calls:

\[
dnf_i = \begin{cases} 
nf_i & \text{if } RC = \emptyset \\
nf_i \ast ri_j & \text{otherwise} \end{cases}
\]

where

- \( RC = \text{set of remote components called by component } i \);
- \( ri_j = \begin{cases} 
nfc_{i, j} & \text{if the call is nonavailable} \\
nfc_{i, j} \ast dnf_j & \text{otherwise} \end{cases} \)

Although this formula covers a wide range of cases, it does not cover all the possible remote call chains. Nevertheless, the following formula calculates the non-failure probability of the first component of a network of remote calls—for instance 4.2(f):

\[
dnf_i = \begin{cases} 
nf_i & \text{if } RC = \emptyset \\
nf_i \ast \prod_{j \in RC} ri_j & \text{otherwise} \end{cases}
\]

where

- \( RC = \text{set of remote components called by component } i \),
- \( ri_j = \begin{cases} 
nfc_{i, j} & \text{if the call is nonavailable} \\
nfc_{i, j} \ast dnf_j & \text{otherwise} \end{cases} \)
4.2.2.3 Maintainability Analysis

In terms of maintainability, a statically bound call is more susceptible to changes being performed in the called component. If a call discovers its parameters or network address at run-time, the calling component does not need modification and therefore the overall system is easier to maintain. We include here changes to the called components and changes to the interface with the communication infrastructure.

Let us consider the simple case of a remote call between distributed components and the following definitions:

- \( tbc_i \) is the mean time between changes of component \( i \) if no remote calls are made.
- \( dtbc_i \) is the mean time between changes of component \( i \) if remote calls can be made—if no remote calls are made, \( dtbc_i = tbc_i \).
- \( tbcc_{i,j} \) is the mean time between changes of the communication infrastructure between components \( i \) and \( j \).

Then, if component \( i \) makes a remote call to component \( j \), its mean time between changes is \( dtbc_i = \min\{tbc_i, tbcc_{i,j}, dtbc_j\} = \min\{tbc_i, tbcc_{i,j}, tbc_j\} \) if the call was statically bound, since component \( i \) needs to be changed if any of the three events occur; and \( dtbc_i = tbc_i \) if the call was dynamically bound, since component \( i \) does not need to be changed if changes are made on the communication infrastructure and/or on component \( j \). Thus, if the call is statically bound, the value of \( dtbc_i \) is less than or at most equal to the value of \( dtbc_j \) when the call is dynamically bound.

If the remote calls are chained as shown in Figures 4.3(a) and 4.3(b), the mean time between changes of component 1 is \( dtbc_1 = \min\{tbc_1, tbcc_{1,2}, dtbc_2\} \) since in both cases the chain begins with a statically bound call. However, \( dtbc_2 \) is different in each case. In the case of Figure 4.3(a) \( dtbc_2 = \min\{tbc_2, tbcc_{2,3}, dtbc_3\} = \min\{tbc_2, tbcc_{2,3}, tbc_3\} \), since the second call is statically bound. In the case of Figure 4.3(b) \( dtbc_2 = tbc_2 \), since the second call is dynamically bound.

In the case of a chain of calls, the inclusion of a dynamically bound call provides a higher mean time between changes to the initial caller. Furthermore, if the chain
Figure 4.3: Remote calls among distributed components (maintainability)
starts with a dynamically bound call [Figures 4.3(c) and 4.3(d)], the mean time between changes could be even higher, since the rest of the chain is decoupled. Therefore, in the cases of Figures 4.3(c) and 4.3(d) the time between changes of component $1$ is $dtbc_1 = tbc_1$.

Hence, if the first call is dynamically bound, $dtbc_1 = tbc_1$ regardless of the second call. The following scale presents the four cases of chains discussed above from worst to best:

- case of Figure 4.3(a)
- case of Figure 4.3(b)
- cases of Figures 4.3(c) and 4.3(d)

This analysis can be extended to deal with the general case—Figure 4.3(e). The following formula calculates the mean time between changes of the first component of a linear chain of remote calls:

$$dtbc_i = \begin{cases} tbc_i & \text{if } SC = \emptyset \\
\min\{tbc_i, tbc_{i,j}, dtbc_j\} & \text{otherwise} \end{cases}$$

where $SC$ is the set of remote components statically called by component $i$.

Although this formula covers a wide range of cases, it does not cover all the possible remote call chains. Nevertheless, the following formula calculates the mean time between changes of the first component of a non-linear network of remote calls—see for example, Figure 4.3(f):

$$dtbc_i = \begin{cases} tbc_i & \text{if } SC = \emptyset \\
\min\{tbc_i, m_{tbc}, dtbc\} & \text{otherwise} \end{cases}$$

where:

- $SC = \text{set of remote components statically called by component } i$
- $m_{tbc} = \min\{tbc_{i,j} | j \in SC\}$
- $m_{dtbc} = \min\{dtbc_j | j \in SC\}$
4.2.3 Summary

Of the several attributes proposed by the ISO 9126-1 quality model [2001], we have chosen to concentrate on the ones we deem more relevant for enterprise DIS, namely efficiency, reliability and maintainability. Other attributes such as portability may also be significant for distributed systems, but they are outside the scope of this project. We have also identified the design attributes that are likely to have a significant impact on those quality attributes of interest, and discussed their relationships as part of the intuitive modelling.

4.3 Formal Model

4.3.1 Measurement Goals

Our main objective is to develop design measures that can be used as indicators of quality. It is widely acknowledged in Software Engineering that early design decisions have a deep impact on the implementation of software and thus, on the ultimate quality of the software as an operational entity.

Quality, as discussed previously, is a complex and multidimensional concept, presenting many facets embodied by quality attributes. Therefore, our derived measurement goals are focused on specific quality attributes that we have argued are significantly affected by the particular nature of DIS.

Consequently, the following measurement goals are defined:

- **Goal 1.1.**
  - *Object of study.* Structural artefacts of DIS.
  - *Purpose.* Evaluation and understanding.
  - *Quality focus.* Efficiency of implemented DIS in testing or operation.
  - *Viewpoint.* Software Engineer.
  - *Environment.* Enterprise DIS.

- **Goal 1.2.**
○ Object of study. Behavioural artefacts of DIS.

○ Purpose. Evaluation and understanding.

○ Quality focus. Efficiency of implemented DIS in testing or operation.

○ Viewpoint. Software Engineer.

○ Environment. Enterprise DIS.

□ Goal 2.1.

○ Object of study. Structural artefacts of DIS.

○ Purpose. Evaluation and understanding.

○ Quality focus. Reliability of implemented DIS in testing or operation.

○ Viewpoint. Software Engineer.

○ Environment. Enterprise DIS.

□ Goal 2.2.

○ Object of study. Behavioural artefacts of DIS.

○ Purpose. Evaluation and understanding.

○ Quality focus. Reliability of implemented DIS in testing or operation.

○ Viewpoint. Software Engineer.

○ Environment. DIS developed by enterprises.

□ Goal 3.1.

○ Object of study. Structural artefacts of DIS.

○ Purpose. Evaluation and understanding.

○ Quality focus. Maintainability of implemented DIS in testing or operation.

○ Viewpoint. Software Engineer.

○ Environment. DIS developed by enterprises.

□ Goal 3.2.

○ Object of study. Behavioural artefacts of DIS.
CHAPTER 4. MEASUREMENT MODELS

- *Purpose.* Evaluation and understanding.
- *Quality focus.* Maintainability of implemented DIS in testing or operation.
- *Viewpoint.* Software Engineer.
- *Environment.* DIS developed by enterprises.

### 4.3.2 Experimental Hypotheses

From the above discussion of the intuitive theories about the relation between DIS design attributes and quality attributes, the following hypotheses can be elucidated:

- **Hypothesis 1.** The structural coupling of a distributed cluster affects its extrinsic efficiency.

- **Hypothesis 2.** The structural coupling of a distributed cluster affects its extrinsic reliability.

- **Hypothesis 3.** The structural coupling of a distributed cluster affects its extrinsic maintainability.

- **Hypothesis 4.** The behavioural coupling of a distributed cluster influences its extrinsic efficiency.

- **Hypothesis 5.** The behavioural coupling of a distributed cluster influences its extrinsic reliability.

- **Hypothesis 6.** The behavioural coupling of a distributed cluster influences its extrinsic maintainability.

These hypotheses establish explicit links between design attributes and quality attributes, that are necessary to define measures that are somewhat supported by an underlying theory to be confirmed or dis-confirmed.

### 4.3.3 Entity Abstractions

We have defined goals to measure attributes from two different design artefacts: a *structural* artefact and a *behavioural* artefact. Although we could have reused
some existing representations such as UML diagrams, we decided to define our own abstractions to represent these artefacts for two reasons:

- our approach is generic, thus it is applicable to a wider range of situations (for example OO and non-OO DIS) and

- our approach is formal, which facilitates the precise definition of our measures and their theoretical validation.

In this dissertation, we are interested in the definition of measures for design artefacts, so we assume that the design already exists. It is the designer’s responsibility to write design artefacts in such a way as to represent the modelled system accurately.

4.3.3.1 Structure

The structure of an enterprise DIS is abstracted by considering the executable software components of the system as the fundamental units and the dependencies among them. A design decision unique to DIS is the allocation of components to the nodes of the computer network, and possibly the nodes to different sub-networks (e.g. LAN, WAN, wireless). Consequently, the structure of a DIS should also reflect this, that is, the structure should contain information about the physical clustering of components.

Therefore, a cluster is typically a group of interdependent components organised around some technological platform. However, there are other alternatives when clustering, and cluster boundaries may be established according to different criteria. For example, a cluster may:

- contain components supporting an enterprise process, or closely related parts of a process;

- support the operations of some enterprise unit (i.e. a section, division or department);

- support the business process of the enterprise at some geographical location.
It should be noted that a cluster may contain just a single component.

The structure of a DIS will be represented by a graph-based abstraction. This approach is commonly used in the literature to model software artefacts [Briand et al., 1996]. A DIS structure $S$ is represented as a 4-tuple $S = < CO, PD, ID, CL >$, where

- $CO$ is a set of (distributed) components,
- $PD \subseteq CO \times CO$ is a set of inter-cluster processing dependencies,
- $ID \subseteq CO \times CO$ is a set of inter-cluster informational dependencies, and
- $CL$ is a set of (distributed) clusters.

A cluster $cl = < CO_{cl}, PD_{cl}, ID_{cl} >$ is a sub-system of $S$ such that

- $CO_{cl} \subseteq CO$ is a set of (distributed) components,
- $PD_{cl} \subseteq CO_{cl} \times CO_{cl}$ is a set of intra-cluster processing dependencies,
- $ID_{cl} \subseteq CO_{cl} \times CO_{cl}$ is a set of intra-cluster informational dependencies, and
- $PD_{cl} \cap PD = ID_{cl} \cap ID = \emptyset$.

It is assumed $CO$ is partitioned into the sets of cluster components:

$$\forall co \in CO(\exists cl \in CL(cl = < CO_{cl}, PD_{cl}, ID_{cl} > \text{ and } co \in CO_{cl}) \text{ and } \forall cl_i, cl_j \in CL(cl_i = < CO_{cl_i}, PD_{cl_i}, ID_{cl_i} > \text{ and } cl_j = < CO_{cl_j}, PD_{cl_j}, ID_{cl_j} > \text{ and } CO_{cl_i} \cap CO_{cl_j} = \emptyset)$$

It should be also noted that inter and intra cluster dependencies are also inter component dependencies, and that processing dependencies are represented by full lines whereas informational dependencies are represented by dashed lines.

We illustrate these concepts by means of the DIS structure $S = < CO, PD, ID, CL >$ presented in Figure 4.4, where

- $CO = \{ a, b, c, d, e, f, g, h, i, j, k, l, m \}$,
- $PD = \{ < b, f >, < c, b >, < d, f >, < h, i > \}$,
- $ID = \emptyset$ and
Figure 4.4: Sample DIS Structure

- CL = \{z, y, x\}, with
  - z = <a, b, h>, \{<h, a>, <b, a>\},
  - y = <f, i, j, k, l>, \{<f, k>, <i, j>, <k, j>, <k, l>, <f, i>\} and
  - x = <c, d, e, g, m>, \{<c, g>, <e, g>, <g, m>, <c, d>\}.

4.3.3.2 Behaviour

Although, the abstraction presented above is adequate to represent the static structure of a DIS, it is not sufficient to model dynamic behavioural aspects. Hence, we extend the model to capture behavioural aspects, in the distributed computing sense, considering distributed cooperative scenarios. A (distributed) cooperative scenario is a "society" of components—not necessarily belonging to the same structural cluster—that interact dynamically to accomplish some collaborative behaviour when operating. Cooperative scenarios are partial descriptions that combined together provide a more complete view of how a DIS may be expected to behave. This abstraction emphasises the time ordering of the interactions by using sequence numbers. Generally, a (distributed) component will be part of several cooperative scenarios, and the behaviour of a DIS will be represented by several cooperative scenarios.

For instance, let us consider the enterprise DIS of a large university to illustrate
that a component is usually part of several cooperative scenarios, and that several scenarios are necessary to describe the behaviour partially:

- As part of scenario A, the application in charge of course enrolments interacts with the application responsible for finances to produce invoices, and with the application in charge of resources administration to allocate the students to specific classes. The application responsible for resources administration, in turn interacts with the application that supports staff management to allocate staff to specific classes.

- As part of scenario B, when the event of new student occurs, the application in charge of program enrolments propagates the information of this event, to the application responsible of course enrolments, to the library sub-system and to IT services support application.

- As part of scenario C, the application in charge of course enrolments and the application responsible of program enrolments interact with the academic management sub-system to update any academic change such as a change in the name of a new program or the creation of a new elective course.

The behaviour of a DIS is also represented by a graph-based abstraction. Formally, a DIS behaviour is 5-tuple \( B = \langle CO, CS, I, ch, s \rangle \), where

- \( CO \) is a set of components;
- \( CS \) is a set of cooperative scenarios;
- \( I \) is a set of remote interactions, \( I = \{ \langle co_1, co_2, cs \rangle | co_1 \in CO \land co_2 \in CO \land cs \in CS \} \);
- \( ch \) is a typifying function which characterises each interaction with a type, \( ch : I \rightarrow T \), where \( T \) denotes the set of interaction types (the elements of \( T \) are listed and described in Table 4.2);
- \( s \) is a sequencing function, \( s : I \rightarrow \{1,2,3,\ldots\} \) which assigns a positive number to each interaction.
Within each scenario, interactions are numbered sequentially starting from 1 to indicate the order in which they are executed. To show concurrency within a scenario, the same number can be assigned to parallel interactions.

Figure 4.5 shows an example of a DIS behaviour \( B = \{CO, CS, I, ch, s\} \), where:

- \( CO = \{a, b, c, d, e, f, g\} \);
- \( CS = \{x, y, z\} \);
- \( I = \{a, b, c, d, e, f, g, y, x\} \);
- \( s(a, b) = s(a, f, y) = s(f, e, x) = 1, s(b, c) = s(a, d, y) = s(f, d, x) = 2, s(c, d, z) = s(d, g, y) = s(f, c, x) = 3, s(a, e, y) = 4 \); and
- \( ch(a, b, z) = ch(a, f, y) = ch(b, c, z) = ch(a, d, y) = ch(f, e, x) = ch(f, d, x) = ch(f, c, x) = SACS, ch(c, d, z) = ch(a, e, y) = SACD, ch(d, c, y) = ch(d, g, y) = ANCS \).

### 4.4 Summary

This chapter has reported the application of the two first stages of the research method. An informal model based on our intuitive analysis of the problem domain has been described in the first part. The collection of intuitions and insights of the informal model have been consolidated into a formal model in the second part. Next chapter presents the results of the following stage, the formal definition of software measures and their theoretical validation.
## Table 4.2: The elements of the set $I'$ (interaction types)

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SACS</td>
<td>Synchronous, Available, Conversational and Statically bound</td>
</tr>
<tr>
<td>SACD</td>
<td>Synchronous, Available, Conversational and Dynamically bound</td>
</tr>
<tr>
<td>SANS</td>
<td>Synchronous, Available, Non-conversational and Statically bound</td>
</tr>
<tr>
<td>SAND</td>
<td>Synchronous, Available, Non-conversational and Dynamically bound</td>
</tr>
<tr>
<td>SNCS</td>
<td>Synchronous, Non-available, Conversational and Statically bound</td>
</tr>
<tr>
<td>SNCD</td>
<td>Synchronous, Non-available, Conversational and Dynamically bound</td>
</tr>
<tr>
<td>SNNS</td>
<td>Synchronous, Non-available, Non-conversational and Statically bound</td>
</tr>
<tr>
<td>SNND</td>
<td>Synchronous, Non-available, Non-conversational and Dynamically bound</td>
</tr>
<tr>
<td>AACS</td>
<td>Asynchronous, Available, Conversational and Statically bound</td>
</tr>
<tr>
<td>AACD</td>
<td>Asynchronous, Available, Conversational and Dynamically bound</td>
</tr>
<tr>
<td>AANS</td>
<td>Asynchronous, Available, Non-conversational and Statically bound</td>
</tr>
<tr>
<td>AAND</td>
<td>Asynchronous, Available, Non-conversational and Dynamically bound</td>
</tr>
<tr>
<td>ANCS</td>
<td>Asynchronous, Non-available, Conversational and Statically bound</td>
</tr>
<tr>
<td>ANCD</td>
<td>Asynchronous, Non-available, Conversational and Dynamically bound</td>
</tr>
<tr>
<td>ANNS</td>
<td>Asynchronous, Non-available, Non-conversational and Statically bound</td>
</tr>
<tr>
<td>ANND</td>
<td>Asynchronous, Non-available, Non-conversational and Dynamically bound</td>
</tr>
</tbody>
</table>
Figure 4.5: Sample DIS Behaviour
Chapter 5

Measure Definition and Theoretical Validation

5.1 Introduction

After the description of the informal and formal models in the previous chapter, we present in this chapter the formal definition and theoretical validation of measures for the attributes of design artefacts. We start by introducing briefly the framework used for the definition and validation of the measures (Section 5.2). The structural measures are defined and validated in section 5.3. The definition and validation of the behavioural measures is put forward in section 5.4.

5.2 Distance Framework

The distance-based approach presents a constructive procedure that defines software measures that are theoretically valid according to the prescriptions of measurement theory [Poels and Dedene, 2000]. This section summarizes the basic concepts used here to make the chapter self-contained (for more details refer to chapter 3). The distance-based measure construction process consists of five major steps:

1. For the set of software entities $E$ and for the internal attribute $a$, select a set of software artefacts $M$ that can be used as measurement abstractions to
2. Define a set $T$ of elementary transformation types on $M$ that is constructively and inverse constructively complete to model the conceptual distances between measurement abstractions,

3. Quantify distances between measurement abstractions defining a metric $\delta : M \times M \to \mathbb{R}$ such that $(M, \delta)$ is a metric space.

4. Select a reference model $\tau \in M$ that is the software measurement abstraction for which it holds that for all $e \in E$ with $\alpha(e) = \tau$, $e$ has the lowest value of $a$.

5. Define a function $\mu : E \to \mathbb{R}$ such that for all $e \in E$, $\mu(e) = \delta(\alpha(e), \tau)$ which is a measure of distance from $\alpha(e)$ to $\tau$.

After the generic definition of $T$ and $(M, \delta)$ in the Section 5.2.1, steps 2 and 3 will be merged in the remainder of this chapter.

5.2.1 A metric space for sets

The majority of the abstraction functions defined in subsequent sections map software entities into sets. These set functions are the measurement abstractions of the distance framework. This framework requires that a metric space (with additive segments) is defined for the set of such abstractions. Hence, to avoid repetition, we need a metric space to quantify the distance between two generic sets, regardless of the elements they contain.

Assume that $G$ is the universe of generic elements used to build the set of abstractions of the software entities in a set $E$. The function $\alpha : E \to \mathcal{P}(G)$ maps each $e \in E$ into a subset of $G$. The metric space is constructed for $\mathcal{P}(G)^1$, so the distance between any two subsets of $G$ is measured. If the set of software entity abstractions is a set of sets, then a generic definition of a constructively complete and inverse constructively complete set $T$ of atomic transformations on sets can also be given. Let $G$ be a set, then $T = \{\theta_1, \theta_2\}$ where

\[\forall m \in \mathcal{P}(G) : \theta_1(m) = m \cup \{g\}, \text{ with } g \in G\]

\(^1\mathcal{P}(G)\) denotes the power set of $F$, i.e. the set of all subsets of $F$. 

84
\[ \forall m \in P(G) : \theta_2(m) = m - \{g\}, \text{ with } g \in G \]

Starting from the empty set \(\emptyset\), which is the initial abstraction \(m_0 \in P(G)\), every set \(m \in P(G)\) can be constructed by applying to \(m_0\) a finite sequence of transformations \(\theta_1\). Likewise, every \(m \in P(G)\) can be taken to \(m_0\) by a finite sequence of transformations \(\theta_2\).

Assume that we have to transform \(m_1 \in P(G)\) into \(m_2 \in P(G)\). As exactly one elementary transformation will be needed for each element of \(G\) that is contained in either \(m_1\) or \(m_2\)—but not in both sets—the length of one the shortest sequences of elementary transformations taking \(m_1\) to \(m_2\) is equal to the cardinality of the symmetric difference between \(m_1\) and \(m_2\). Then, the metric \(\delta_{SET} : P(G) \times P(G) \to \mathbb{R}\) is defined as:

\[
\delta_{SET}(m_1, m_2) = |m_1 \Delta m_2| = |(m_1 - m_2) \cup (m_2 - m_1)| = |m_1 - m_2| + |m_2 - m_1| \quad (5.1)
\]

Although this number may be multiplied by any positive real number \(k\), it is assumed here that \(k = 1\). According to Suppes et al. [1989] this metric takes the form of a symmetric difference model.

### 5.2.2 An example

As an illustration of distance-based software measurement, the measure number of fields (NF)\(^2\)—proposed by Plattini et al. [2001a]—is defined (and validated) to characterise the complexity of relational database tables.

Let \(E\) be a set of tables associated with some Universe of Discourse (i.e. domain or model) and let \(F\) be the Universe of Fields relevant to \(E\). The application of the distance-based approach results in:

- The definition of a set of measurement abstractions \(P(F)\) and a function \(\alpha_{NF} : E \to P(F)\) that captures table complexity: \(\forall e \in E : \alpha(e)_{NF} = \{ f \in F | f \text{ is a field of } e \}\). The measurement abstraction \(m\) for a table \(e\) in \(E\) is the set of fields in \(F\) that are part of \(e\).

\(^2\)The original measure is actually number of attributes (NA). The name was modified to avoid confusion between attributes of relational tables and attributes of software entities.
The definition of a set $T$ of elementary transformations on $P(F)$ that is constructively and inverse constructively complete. Since the elements of $P(F)$ are sets, $T$ only contains a transformation that adds a field to a set and a transformation that removes a field from a set. Thus, $T = \{ \theta_{1NF}, \theta_{2NF} \}$ where

- $\forall m \in P(F) : \theta_{1NF}(m) = m \cup \{ f \}$, with $f \in F$
- $\forall m \in P(F) : \theta_{2NF}(m) = m - \{ f \}$, with $f \in F$

The definition of a metric space $(P(F), \delta_{NF})$ where $\forall m_1, m_2 \in P(F) : \delta_{NF}(m_1, m_2) = |m_1 - m_2| + |m_2 - m_1|$.

The definition of the reference abstraction $\tau \in P(F)$ for table complexity. A table with no fields is the table with the minimal complexity, hence $\tau_{NF} = \emptyset$.

The definition of a table complexity measure $\mu_{NF} : E \rightarrow \mathbb{R}$ where $\forall e \in E : \mu_{NF}(e) = \delta_{NF}(\alpha_{NF}(e), \tau_{NF}) = |\alpha_{NF}(e) - \emptyset| + |\emptyset - \alpha_{NF}(e)| = |\alpha_{NF}(e)|$. This measure therefore counts the number of fields of table $e$.

### 5.3 Structural Coupling

Due to the importance of the consequences, demarcating cluster boundaries is a central issue—as we argued previously—when designing the structure of a DIS. Therefore, inter-cluster coupling is seen as a key attribute to assess. This type of coupling is determined by the relationships (dependencies) between clusters in the structure of a DIS.

Each type of relationship defines a sub-attribute of coupling: informational coupling and processing coupling. Because of their potentially different impact on the software quality attributes of interest, the two forms of structural coupling are considered separately. Coupling can also be evaluated from an inbound or an outbound viewpoint. Again, potentially different impacts on the quality attributes of interest are expected. Therefore, four different structural coupling sub-attributes are distinguished:

- inbound informational (II) coupling;
5.3. STRUCTURAL COUPLING

- inbound processing (IP) coupling;
- outbound informational (OI) coupling;
- outbound processing (OP) coupling.

Furthermore, this artefact (i.e. the structure of DIS) contains elements of three different kinds: the *cluster*, the *component*, and the *dependency*. Therefore, the measurement of the four coupling sub-attributes can be analysed from these three different perspectives. The following sub-sections define and validate theoretically the resulting 12 measures.

In order to exemplify the process we will use the structure of the DIS shown in Figure 5.1. This system has three clusters: one for each organisational unit of a wholesale bookstore. Each cluster is composed of a three-tier client/server application and a message gateway component to interact with other clusters. C/S applications have three components: the user interface, the business logic server and the data access server.

5.3.1 Cluster Perspective

5.3.1.1 Measurement Abstractions

Each of the structural coupling sub-attributes requires a suitable measurement abstraction to be defined from this perspective. The abstraction function \( \alpha \) for a particular coupling sub-attribute returns, given a distributed cluster \( e \), the set of clusters in \( S \) to which \( e \) is coupled. Note the elements of the abstractions are sets (of clusters). The abstraction functions are defined in Table 5.1. Indirect dependencies between clusters are not taken into account by any of these abstractions.

Let \( E' \) be the set of components of some structural abstraction \( S \). A cluster \( e \in E \) is coupled to a cluster \( f \in E \) if \( \exists e', f' \in E' \) such that:

- \( e' \) is coupled to \( f' \), and
- \( e' \) is part of \( e \) and \( f' \) is part of \( f \), and
- \( e \neq f \).

In the example of Figure 5.1, \( \alpha_{CLOP}(Sales) = \{HO,WH\} \) and \( \alpha_{CLOP}(WH) = \{Sales\} \).
\[ S = \langle \text{CO}, \text{PD}, \text{ID}, \text{CL} \rangle, \text{ where} \]

- \( \text{CO} = \{ \text{ho_ui}, \text{wh_ui}, \text{s_ui}, \text{ho_bs}, \text{wh_bs}, \text{s_bs}, \text{wh_ds}, \text{ho_ds}, \text{s_ds}, \text{wh_ma}, \text{ho_ma}, \text{s_ma} \} \),

- \( \text{PD} = \emptyset \),

- \( \text{ID} = \{ \langle \text{ho_ma}, \text{s_ma} \rangle, \langle \text{s_ma}, \text{ho_ma} \rangle, \langle \text{wh_ma}, \text{s_ma} \rangle, \langle \text{s_ma}, \text{wh_ma} \rangle, \langle \text{ho_ma}, \text{wh_ma} \rangle \} \), and

- \( \text{CL} = \{ \text{WH, HO, Sales} \} \), with
  - \( \text{WH} = \langle \{ \text{wh_ma}, \text{wh_bs}, \text{wh_ds}, \text{wh_ui} \}, \{ \langle \text{wh_ui}, \text{wh_bs} \rangle, \langle \text{wh_bs}, \text{wh_ds} \rangle, \langle \text{wh_ds}, \text{wh_ma} \rangle, \langle \text{wh_ma}, \text{wh_bs} \rangle \}, \emptyset \rangle \),
  - \( \text{HO} = \langle \{ \text{ho_ma}, \text{ho_bs}, \text{ho_ds}, \text{s_ui} \}, \{ \langle \text{ho_ui}, \text{ho_bs} \rangle, \langle \text{ho_bs}, \text{ho_ds} \rangle, \langle \text{ho_ds}, \text{ho_ma} \rangle, \langle \text{ho_ma}, \text{ho_bs} \rangle \}, \emptyset \rangle \),
  - \( \text{Sales} = \langle \{ \text{s_ma}, \text{s_bs}, \text{s_ds}, \text{s_ui} \}, \{ \langle \text{s_ui}, \text{s_bs} \rangle, \langle \text{s_bs}, \text{s_ds} \rangle, \langle \text{s_ds}, \text{s_ma} \rangle, \langle \text{s_ma}, \text{s_bs} \rangle \}, \emptyset \rangle \).

Figure 5.1: An example of the structure of a DIS
Let $E$ be the set of clusters associated with some Universe of Discourse (i.e. a structural model $S$). Then $\forall e \in E$:

- $\alpha_{\text{CLOP}}(e) = \{ f \in E | f \text{ is coupled by OP dependencies to } e \}$
- $\alpha_{\text{CLOI}}(e) = \{ f \in E | f \text{ is coupled by OI dependencies to } e \}$
- $\alpha_{\text{CLIP}}(e) = \{ f \in E | f \text{ is coupled by IP dependencies to } e \}$
- $\alpha_{\text{CLIH}}(e) = \{ f \in E | f \text{ is coupled by II dependencies to } e \}$

Table 5.1: Structural Coupling Measurement Abstractions (Cluster Perspective)

### 5.3.1.2 Elementary transformations and metric spaces

All measurement abstractions—i.e. the elements of $M$—are sets (of clusters in this case). We therefore need to model and quantify distances between sets. A generic metric space for sets, based on the symmetric difference model, has been defined before in section 5.2.1. Elementary transformations of $T$ either add or remove an element from a set. As exactly one elementary transformation will be needed for each element of $E$ that is contained in either $m$ or $m'$, but not in both sets, the distance is equal to the cardinality of the symmetric difference between $m$ and $m'$.

In the example of Figure 5.1, $\delta_{\text{SET}}(\alpha_{\text{CLOP}}(Sales), \alpha_{\text{CLOP}}(WH)) = |\{HO, WH\}| + |\{Sales\}| = 2 + 1 = 3$

### 5.3.1.3 Reference abstractions

There might be clusters that are not coupled to any other cluster in the structure of a DIS. Therefore empty sets are chosen as reference abstractions for OP, OI, IP, and II coupling. Formally,

- $\forall e \in E : \tau_{\text{CLOP}}(e) = \tau_{\text{CLOI}}(e) = \tau_{\text{CLIP}}(e) = \tau_{\text{CLIH}}(e) = \emptyset$

Table 5.2: Structural Coupling Reference Abstractions (Cluster Perspective)
5.3.1.4 Measures definition

Given the choices regarding software measurement abstractions, reference abstractions, and metric made above, these structural coupling measures return, for a cluster e, the count of couples with other clusters due to dependencies (of the relevant type) in the structure of a DIS. In other words, these measures capture the conceptual distance between the measurement abstraction and the reference abstraction. To avoid repetition, we present a *generic* measure definition in Table 5.3

\[
\forall e \in E : \mu_{cl}(e) = \delta_{set}(\alpha(e), \tau(e)) \text{ measures } a \text{ of } e, \text{ where } \\
\mu, \alpha, \tau, \text{ and } a \text{ are tabulated in Table 5.4}
\]

<table>
<thead>
<tr>
<th>Attribute (a)</th>
<th>Measurement Abstraction ((\alpha))</th>
<th>Reference Abstraction ((\tau))</th>
<th>Measure ((\mu))</th>
</tr>
</thead>
<tbody>
<tr>
<td>OP coupling</td>
<td>(\alpha_{CLOP})</td>
<td>(\tau_{CLOP})</td>
<td>CLOP</td>
</tr>
<tr>
<td>OI coupling</td>
<td>(\alpha_{CLOI})</td>
<td>(\tau_{CLOI})</td>
<td>CLOI</td>
</tr>
<tr>
<td>IP coupling</td>
<td>(\alpha_{CLIP})</td>
<td>(\tau_{CLIP})</td>
<td>CLIP</td>
</tr>
<tr>
<td>II coupling</td>
<td>(\alpha_{CLII})</td>
<td>(\tau_{CLII})</td>
<td>CLII</td>
</tr>
</tbody>
</table>

Table 5.3: Structural Coupling Measures Definition (Cluster Perspective)

In our example (Figure 5.1), \(\mu_{CLOP}(HO) = |\{Sales, WH\}| = 2\) and \(\mu_{CLOP}(WH) = |\{Sales\}| = 1\).

5.3.2 Component perspective

5.3.2.1 Measurement Abstractions

From this perspective, each of the structural coupling sub-attributes also requires a different measurement abstraction to be defined. The abstraction function \(\alpha\) of a specific coupling sub-attribute returns, given a distributed cluster \(e\) of the structure \(S\), a set of components that:
5.3. STRUCTURAL COUPLING

Let \( E \) be the set of clusters, and \( E' \) be the set of components associated with some Universe of Discourse (i.e., a structural model \( S \)). Then \( \forall e \in E: \)

\[
\begin{align*}
\alpha_{\text{COOP}}(e) &= \{ e' \in E' | e' \text{ is coupled by OP dependencies to } f \in E \land e' \text{ is part of } e \} \\
\alpha_{\text{OI}}(e) &= \{ e' \in E' | e' \text{ is coupled by OI dependencies to } f \in E \land e' \text{ is part of } e \} \\
\alpha_{\text{IP}}(e) &= \{ e' \in E' | e' \text{ is coupled by IP dependencies to } f \in E \land e' \text{ is part of } e \} \\
\alpha_{\text{II}}(e) &= \{ e' \in E' | e' \text{ is coupled by II dependencies to } f \in E \land e' \text{ is part of } e \}
\end{align*}
\]

Table 5.5: Structural Coupling Measurement Abstractions (Component Perspective)

- \( e \) are part of \( e \), and
- \( e \) are coupled to components of other clusters.

The elements of each abstraction are sets (of components in this case). The measurement abstraction functions are defined in Table 5.5. Indirect coupling between components is not taken into account by any of these abstractions.

A component \( e' \in E' \) of cluster \( e \in E \) is coupled to a cluster \( f \in E \), if \( \exists f' \in E' \) such that:

- \( e' \) is coupled to \( f' \), and
- \( e' \) is part of \( e \) and \( f' \) is part of \( f \), and
- \( e \neq f \).

In the example of Figure 5.1, \( \alpha_{\text{COOP}}(HO) = \{ ho_ma \} \) and \( \alpha_{\text{COOP}}(Sales) = \{ s_ma \} \).

5.3.2.2 Elementary transformations and metric spaces

The elements of \( M \) are also sets (of components in this case). Therefore, the generic metric space for sets defined in Section 5.2.1 can be used.

5.3.2.3 Reference abstractions

There might be clusters with no components coupled to other cluster components in the structure of a DIS. Therefore empty sets are chosen as reference abstractions for OP, OI, IP, and II coupling. Formally,
<table>
<thead>
<tr>
<th>Attribute (a)</th>
<th>Measurement Abstraction (α)</th>
<th>Reference Abstraction (τ)</th>
<th>Measure (μ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OP coupling</td>
<td>α_{COOP}</td>
<td>τ_{COOP}</td>
<td>COOP</td>
</tr>
<tr>
<td>OI coupling</td>
<td>α_{COII}</td>
<td>τ_{COII}</td>
<td>COII</td>
</tr>
<tr>
<td>IP coupling</td>
<td>α_{COIP}</td>
<td>τ_{COIP}</td>
<td>COIP</td>
</tr>
<tr>
<td>II coupling</td>
<td>α_{COII}</td>
<td>τ_{COII}</td>
<td>COII</td>
</tr>
</tbody>
</table>

Table 5.8: Structural Coupling Measures Legend (Component Perspective)

∀ε ∈ E : τ_{COOP}(ε) = τ_{COII}(ε) = τ_{COIP}(ε) = τ_{COII}(ε) = ∅

Table 5.6: Structural Coupling Reference Abstractions (Component Perspective)

### 5.3.2.4 Measures definition

Given the choices regarding software measurement abstractions, reference abstractions, and metric made above, these structural coupling measures return the number of components coupled by inter-cluster dependencies (of the relevant type) for a cluster ε in the structure of DIS. We present a generic measure definition in Table 5.7 to avoid repetitiousness.

∀ε ∈ E : μ_{CO}(ε) = δ_{SET}(α(ε), τ(ε)) measures a of ε, where μ, α, τ, and a are tabulated in Table 5.8

Table 5.7: Structural Coupling Measures Definition (Component Perspective)

In the example of Figure 5.1, \( μ_{COOP}(HO) = |\{ho_ma\}| = 1 \) and \( μ_{COOP}(Sales) = |\{s_ma\}| = 1 \).
Let $E$ be the set of clusters and $E''$ be the set of dependencies associated with some Universe of Discourse (i.e. a structural model $S$). Then $\forall e \in E$:

$$
\alpha_{DOP}(e) = \{e'' \in E'' | e'' \text{ is an OP dependency } \wedge e'' \text{ couples } e \in E \wedge (f \neq e)\}
$$

$$
\alpha_{DOI}(e) = \{e'' \in E'' | e'' \text{ is an OI dependency } \wedge e'' \text{ couples } e \in E \wedge (f \neq e)\}
$$

$$
\alpha_{DIP}(e) = \{e'' \in E'' | e'' \text{ is an IP dependency } \wedge e'' \text{ couples } e \in E \wedge (f \neq e)\}
$$

$$
\alpha_{DIH}(e) = \{e'' \in E'' | e'' \text{ is an II dependency } \wedge e'' \text{ couples } e \in E \wedge (f \neq e)\}
$$

Table 5.9: Structural Coupling Measurement Abstractions (Dependency Perspective)

### 5.3.3 Dependency Perspective

#### 5.3.3.1 Measurement Abstractions

From this perspective, each of the structural coupling sub-attributes requires an appropriate measurement abstraction to be defined as well. The abstraction function $\alpha$ for a specific coupling sub-attribute returns, given a distributed cluster $e$, the set of inter-cluster dependencies of $e$. The elements of each abstraction are also sets (of dependencies in this case). The measurement abstraction functions are defined in Table 5.9.

Let $E'$ be the set of components of the structure $S$ of some DIS. A cluster $e \in E$ is coupled by a dependency $e'' = \langle e', f' \rangle$ to a cluster $f \in E$ if $\exists e, f' \in E'$ such that:

1. $e'$ is part of $e$ and $f'$ is part of $f$, and
2. $e \neq f$.

In the example of Figure 5.1, $\alpha_{DOP}(HO) = \{\langle ho_ma, wh_ma \rangle, \langle ho_ma, s_ma \rangle\}$ and $\alpha_{DOP}(WH) = \{\langle wh_ma, s_ma \rangle\}$.

#### 5.3.3.2 Elementary transformations and metric spaces

The elements of $M$ are also sets (of dependencies in this case). Therefore, the generic metric space for sets defined above in Section 5.2.1 can be used likewise.
### Reference abstractions

There might be clusters not coupled (by dependencies) to other clusters in a DIS structural model. Therefore empty sets are chosen as reference abstractions for OP, OI, IP, and II coupling. Formally,

$$\forall e \in E : \tau_{DOP}(e) = \tau_{DOI}(e) = \tau_{DIP}(e) = \tau_{DII}(e) = \emptyset$$

### Measures definition

To avoid duplication, we present a generic software measure definition in Table 5.11. Given the choices regarding measurement abstractions, reference abstractions, and metric made above, these structural coupling measures return the count of inter-cluster dependencies (of the relevant type) for a cluster $e$ in the structure of a DIS. These measures, therefore, capture the conceptual distance between the measurement abstraction and the reference abstraction.

$$\forall e \in E : \mu_D(e) = \delta_{SET}(\alpha(e), \tau(e))$$ measures $a$ of $e$, where

$\mu, \alpha, \tau$, and $a$ are tabulated in Table 5.12

In the example of Figure 5.1, $\mu_{DOP}(HO) = |\{< ho_ma, wh_ma >, < ho_ma, s_ma >\}| = 2$ and $\mu_{DOP}(WH) = |\{wh_ma, s_ma\}| = 1.$
5.4 Behavioural Coupling

We have seen that the structural design of a DIS introduces some degree of coupling. From the behavioural perspective, the software engineer should select interactions types among components with the lowest possible additional coupling. This selection would avoid the introduction of unnecessary behavioural coupling, while it is consistent with the structure of the system and the underlying middleware infrastructure interfaces. Therefore, interaction among components is considered an important attribute to assess. This type of coupling is determined by the dynamic cooperation between components in the behaviour of a DIS.

Each type of interaction defines a sub-attribute of coupling: synchronisation, availability, conversation, and binding coupling. Because of their potentially different impact on the quality attributes of interest, the four forms of behavioural coupling are considered separately. Furthermore, each type of behavioural coupling can be evaluated from an inbound or outbound viewpoint. Again, potentially different impacts on quality attributes of interest are to be expected, so they are considered individually. Accordingly, we identify 8 behavioural coupling sub-attributes:

- inbound synchronisation (IS) coupling
- inbound availability (IA) coupling
- inbound conversation (IC) coupling
- inbound binding (IB) coupling
- outbound synchronisation (OS) coupling
- outbound availability (OA) coupling
- outbound conversation (OC) coupling
- outbound binding (OB) coupling

The behaviour of a DIS contains elements of three different kinds: the scenario, the component, and the interaction. Therefore, each of the 8 coupling sub-attributes can be measured from three different perspectives.
These three perspectives only take into account direct coupling. Notwithstanding, we contend that indirect interactions coupling is also useful to provide additional insight about the behaviour of a DIS. Consequently, this additional perspective is also considered to capture the 8 behavioural coupling sub-attributes.

In order to illustrate the process we will use the behaviour of the sample DIS shown in Figure 5.2.

### 5.4.1 Component perspective

#### 5.4.1.1 Measurement abstractions

Each of the behavioural coupling sub-attributes requires an appropriate measurement abstraction to be defined. The abstraction function $\alpha$ for a specific coupling sub-attribute returns, given a distributed component $e$, the set of components that are coupled to $e$. The elements of each abstraction are sets (of components in this case). The measurement abstraction functions are defined in Table 5.13. Indirect interactions between components are not taken into account by any of these abstractions.

For example, $\alpha_{\text{cos}}(\text{HoBs}) = \{\text{HoDs, HoMa}\}$ and $\alpha_{\text{cos}}(\text{WhBs}) = \{\text{WhDs}\}$, in the case of Figure 5.2.

#### 5.4.1.2 Elementary transformations and metric space

All measurement abstractions, i.e. the elements of $M$, are sets (of components). We therefore need to model and quantify distances between sets. A generic metric space for sets has been defined before in section 5.2.1. Elementary transformations of $T$ either add or remove an element from a set. As exactly one elementary transformation will be needed for each element of $E$ that is contained in either $m$ or $m'$, but not in both sets, the distance is equal to the cardinality of the symmetric difference between $m$ and $m'$.

In the example of Figure 5.2, $\delta_{\text{set}}(\alpha_{\text{cos}}(\text{HoBs}), \alpha_{\text{cos}}(\text{WhBs})) = |\{\text{HoDs, HoMa}\} - \{\text{WhDs}\}| + |\{\text{WhDs}\} - \{\text{HoDs, HoMa}\}| = 2 + 1 = 3.
B = \langle CO, CS, I, ch, s \rangle, where

- CO = \{HoDs, HoBs, HoUi, HoMa, WhDs, WhBs, WhMa, SDs, SBs, SMA\},

- CS = \{NI, UI\},

- I = \langle HoUi, HoBs, UI\rangle, \langle HoBs, HoDs, UI\rangle, \langle HoUi, HoBs, NI\rangle, \langle HoBs, HoDs, NI\rangle, \langle HoBs, HoMa, NI\rangle, \langle HoMa, WhMa, NI\rangle, \langle HoMa, SMA, NI\rangle, \langle WhMa, WhBs, NI\rangle, \langle SMA, SBs, NI\rangle, \langle WhBs, WhDs, NI\rangle, \langle SBs, SDs, NI\rangle; \langle HoUi, HoBs, UI\rangle\} = \langle HoBs, HoDs, UI\rangle = \langle HoUi, HoBs, NI\rangle = \langle HoBs, HoMa, NI\rangle = \langle HoMa, WhMa, NI\rangle = \langle HoMa, SMA, NI\rangle = \langle WhMa, WhBs, NI\rangle = \langle SMA, SBs, NI\rangle = \langle WhBs, WhDs, NI\rangle = \langle SBs, SDs, NI\rangle = SACS, \langle HoMa, WhMa, NI\rangle = \langle HoMa, SMA, NI\rangle = ANCS; and

- s(\langle HoUi, HoBs, UI\rangle) = s(\langle HoUi, HoBs, NI\rangle) = 1, s(\langle HoBs, HoDs, UI\rangle) = s(\langle HoBs, HoDs, NI\rangle) = 2, s(\langle HoBs, HoMa, NI\rangle) = 3, s(\langle WhMa, WhBs, NI\rangle) = s(\langle SMA, SBs, NI\rangle) = 5, s(\langle WhBs, WhDs, NI\rangle) = s(\langle SBs, SDs, NI\rangle) = 6, s(\langle HoMa, WhMa, ND\rangle) = s(\langle HoMa, SMA, NI\rangle) = 4.

Figure 5.2: An example of the behaviour of a DIS
Let $E$ be the set of components associated with some Universe of Discourse (i.e., a behavioural abstraction $B$). Then $\forall e \in E$:

$$\alpha_{GIS}(e) = \{ f \in E | f \text{ is coupled by IS interactions to } e \}$$

$$\alpha_{GIA}(e) = \{ f \in E | f \text{ is coupled by IA interactions to } e \}$$

$$\alpha_{GIC}(e) = \{ f \in E | f \text{ is coupled by IC interactions to } e \}$$

$$\alpha_{GIB}(e) = \{ f \in E | f \text{ is coupled by IB interactions to } e \}$$

$$\alpha_{GOS}(e) = \{ f \in E | f \text{ is coupled by OS interactions to } e \}$$

$$\alpha_{COA}(e) = \{ f \in E | f \text{ is coupled by OA interactions to } e \}$$

$$\alpha_{COC}(e) = \{ f \in E | f \text{ is coupled by OC interactions to } e \}$$

$$\alpha_{COB}(e) = \{ f \in E | f \text{ is coupled by OB interactions to } e \}$$

| Table 5.13: Behavioural Coupling Measurement Abstractions (Component Perspective) |

### 5.4.1.3 Reference abstractions

There might be components that are not coupled to any other component in the behaviour of a DIS. Therefore, empty sets are chosen as reference abstractions for IS, IA, IC, IB, OS, OA, OC, and OB coupling. Formally,

$$\forall e \in E : \tau_{GIS}(e) = \tau_{GIA}(e) = \tau_{GIC}(e) = \tau_{GIB}(e) = \emptyset$$

$$\forall e \in E : \tau_{GOS}(e) = \tau_{COA}(e) = \tau_{COC}(e) = \tau_{COB}(e) = \emptyset$$

| Table 5.14: Behavioural Coupling Reference Abstractions (Component Perspective) |

### 5.4.1.4 Measures Definition

To avoid reiteration, we present a generic measure definition in Table 5.15. Given the choices regarding software measurement abstractions, reference abstractions, and metric made above, these behavioural coupling measures return for a component $e$, the number of components to which it is coupled by interactions (of the relevant type) in the behaviour of a DIS. That is, these measures capture the
conceptual distance between the measurement abstraction and the reference abstraction.

For instance, $\mu_{cos}(\text{HoBs}) = |\{\text{HoDs, HoMa}\}| = 2$ and $\mu_{cos}(\text{WhBs}) = |\{\text{WhDs}\}| = 1$, in the case of Figure 5.2.

$$\forall e \in E : \mu_c(e) = \delta_{set}(\alpha(e), \tau(e))$$
measures $a$ of $e$, where $\mu, \alpha, \tau$, and $a$ are tabulated in Table 5.16.

Table 5.15: Behavioural Coupling Measures Definition (Component Perspective)

### 5.4.2 Interaction perspective

#### 5.4.2.1 Measurement Abstractions

Each of the behavioural coupling sub-attributes requires an appropriate measurement abstraction to be defined. The abstraction function $\alpha$ for a specific coupling sub-attribute returns, given a distributed component $e$, the set interactions (of the relevant type) of which $e$ is part. The elements of each abstraction are sets of interactions. The measurement abstraction functions are defined in Table 5.17. Note that indirect interactions are not taken into account by any of these abstractions.

In the example of Figure 5.2, $\alpha_{ios}(\text{HoBs}) = \{<\text{HoBs, HoDs, UI}>, <\text{HoBs, HoDs, NI}>, <\text{HoBs, HoMa, NI}>\}$.

#### 5.4.2.2 Elementary transformations and metric spaces

The elements of $M$ are also sets, of interactions in this case. The generic metric space for sets, defined in Section 5.2.1, can be used as well.

#### 5.4.2.3 Reference abstractions

There might be components not coupled (by interactions) to other components in the behaviour of a DIS. Therefore empty sets are chosen as reference abstractions for IS, IA, IC, IB, OS, OA, OC, and OB coupling. Formally,
<table>
<thead>
<tr>
<th>Attribute (a)</th>
<th>Measurement Abstraction (α)</th>
<th>Reference Abstraction (τ)</th>
<th>Measure (μ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IS coupling</td>
<td>α_{GIS}</td>
<td>τ_{GIS}</td>
<td>CIS</td>
</tr>
<tr>
<td>IA coupling</td>
<td>α_{GIA}</td>
<td>τ_{GIA}</td>
<td>CIA</td>
</tr>
<tr>
<td>IC coupling</td>
<td>α_{GIC}</td>
<td>τ_{GIC}</td>
<td>CIC</td>
</tr>
<tr>
<td>IB coupling</td>
<td>α_{GIB}</td>
<td>τ_{GIB}</td>
<td>CIB</td>
</tr>
<tr>
<td>OS coupling</td>
<td>α_{GOS}</td>
<td>τ_{GOS}</td>
<td>COS</td>
</tr>
<tr>
<td>OA coupling</td>
<td>α_{GOA}</td>
<td>τ_{GOA}</td>
<td>COA</td>
</tr>
<tr>
<td>OC coupling</td>
<td>α_{GOC}</td>
<td>τ_{GOC}</td>
<td>COC</td>
</tr>
<tr>
<td>OB coupling</td>
<td>α_{COB}</td>
<td>τ_{COB}</td>
<td>COB</td>
</tr>
</tbody>
</table>

Table 5.16: Behavioural Coupling Measures Legend (Component Perspective)

Let $E$ be the set of components and $E''$ be the set of interactions associated with some Universe of Discourse (i.e. a behavioural abstraction $B$). Then $\forall e \in E$:

$\alpha_{IIS}(e) = \{ e'' \in E'' \mid e \text{ is coupled by } e'' \text{ to } f \in E \land e'' \text{ is an IS interaction} \}$

$\alpha_{IIA}(e) = \{ e'' \in E'' \mid e \text{ is coupled by } e'' \text{ to } f \in E \land e'' \text{ is an IA interaction} \}$

$\alpha_{IIC}(e) = \{ e'' \in E'' \mid e \text{ is coupled by } e'' \text{ to } f \in E \land e'' \text{ is an IC interaction} \}$

$\alpha_{IIB}(e) = \{ e'' \in E'' \mid e \text{ is coupled by } e'' \text{ to } f \in E \land e'' \text{ is an IB interaction} \}$

$\alpha_{IOS}(e) = \{ e'' \in E'' \mid e \text{ is coupled by } e'' \text{ to } f \in E \land e'' \text{ is an OS interaction} \}$

$\alpha_{IOA}(e) = \{ e'' \in E'' \mid e \text{ is coupled by } e'' \text{ to } f \in E \land e'' \text{ is an OA interaction} \}$

$\alpha_{IOC}(e) = \{ e'' \in E'' \mid e \text{ is coupled by } e'' \text{ to } f \in E \land e'' \text{ is an OC interaction} \}$

$\alpha_{IOA}(e) = \{ e'' \in E'' \mid e \text{ is coupled by } e'' \text{ to } f \in E \land e'' \text{ is an OB interaction} \}$

Table 5.17: Behavioural Coupling Measurement Abstractions (Interaction Perspective)
5.4. BEHAVIOURAL COUPLING

\[
\forall e \in E : \tau_{IIc}(e) = \tau_{IIC}(e) = \tau_{HCB}(e) = \emptyset \\
\forall e \in E : \tau_{I03}(e) = \tau_{I0A}(e) = \tau_{I0C}(e) = \tau_{I0B}(e) = \emptyset
\]

Table 5.18: Behavioural Coupling Reference Abstractions (Interaction Perspective)

5.4.2.4 Measures definition

To avoid repetition, we present a generic software measure definition in Table 5.19. Given the choices regarding measurement abstractions, reference abstractions, and metric made above, these behavioural coupling measures return, for a component \(e\), the count of interactions (of the relevant type) of which \(e\) is part in the behaviour of a DIS. Note that indirect interactions between components are not captured by any of these abstractions.

In the example of Figure 5.2, \(\mu_{I03}(HoBs) = |\{<HoBs, HoDs,UI>, <HoBs, HoDs, NI >, < HoBs, HoMa, NI >\}| = 3.

\[
\forall e \in E : \mu_i(e) = \delta_{set}(\alpha(e), \tau(e)) \text{ measures } a \text{ of } e, \text{ where} \\
\text{ } \mu, \alpha, \tau, \text{ and } a \text{ are tabulated in Table 5.20}
\]

Table 5.19: Behavioural Coupling Measures Definition (Interaction Perspective)

5.4.3 Scenario perspective

The rationale behind this perspective is that the higher the number of scenarios of which a component is part, the higher the probability remote interaction during execution.

5.4.3.1 Measurement abstractions

Each of the 8 behavioural coupling sub-attributes requires a suitable measurement abstraction to be defined. The abstraction function \(\alpha\) for a particular coupling sub-attribute returns, given a distributed component \(e\), the set of scenarios of which \(e\)
is part. The abstraction functions are defined in Table 5.21. Note that the elements of the abstractions are sets (of scenarios).

Let $B$ be the behaviour of a DIS. Then, if a distributed component $co \in CO$ is part of a scenario $cs \in CS$, there exists an interaction $i \in I$ such that $i =< co, cd', cs >$ or $i =< cd', co, cs >$, and $cd' \in CO$.

In the example of Figure 5.2, $\alpha_{sos}(HoBs) = \{UI, NI\}$ and $\alpha_{sos}(WhBs) = \{NI\}$.

### 5.4.3.2 Elementary transformations and metric space

The elements of $M$ are also sets, of scenarios in this case. Therefore, the generic metric space for sets defined in Section 5.2.1 can be used as well.

### 5.4.3.3 Reference abstractions

There might be components that are not coupled to any other component in the behaviour of a DIS. Therefore empty sets are chosen as reference abstractions for IS, IA, IC, IB, OS, OA, OC, and OB coupling. Formally,
Let $E$ be the set of components and let $E'$ be the set of scenarios associated with some Universe of Discourse (i.e. the behaviour abstraction $B$). Then $\forall e \in E$:

- $\alpha_{SIS}(e) = \{e' \in E'| e \text{ is coupled by an IS interaction to } f \in E \land (e, f \text{ are part of } e')\}$
- $\alpha_{SIA}(e) = \{e' \in E'| e \text{ is coupled by an IA interaction to } f \in E \land (e, f \text{ are part of } e')\}$
- $\alpha_{SIC}(e) = \{e' \in E'| e \text{ is coupled by an IC interaction to } f \in E \land (e, f \text{ are part of } e')\}$
- $\alpha_{SIB}(e) = \{e' \in E'| e \text{ is coupled by an IB interaction to } f \in E \land (e, f \text{ are part of } e')\}$
- $\alpha_{SOS}(e) = \{e' \in E'| e \text{ is coupled by an OS interaction to } f \in E \land (e, f \text{ are part of } e')\}$
- $\alpha_{SOA}(e) = \{e' \in E'| e \text{ is coupled by an OA interaction to } f \in E \land (e, f \text{ are part of } e')\}$
- $\alpha_{SOC}(e) = \{e' \in E'| e \text{ is coupled by an OC interaction to } f \in E \land (e, f \text{ are part of } e')\}$
- $\alpha_{SOB}(e) = \{e' \in E'| e \text{ is coupled by an OB interaction to } f \in E \land (e, f \text{ are part of } e')\}$

Table 5.21: Behavioural Coupling Measurement Abstractions (Scenario Perspective)
\[ \forall e \in E : \tau_{S1S}(e) = \tau_{S1A}(e) = \tau_{S1C}(e) = \tau_{S1B}(e) = \emptyset \]
\[ \forall e \in E : \tau_{S0S}(e) = \tau_{S0A}(e) = \tau_{S0C}(e) = \tau_{S0B}(e) = \emptyset \]

Table 5.22: Behavioural Coupling Reference Abstractions (Scenario Perspective)

### 5.4.3.4 Measures Definition

Given the choices regarding measurement abstractions, reference abstractions, and metric made above, these behavioural coupling measures return the count of scenarios of which component \( e \) is part due to interactions (of the relevant type) to other components in the behaviour of a DIS. In other words, the measure captures the conceptual distance between the measurement abstraction and the reference abstraction. To avoid repetition, we present a generic software measure definition in Table 5.23.

For instance, \( \mu_{S0S}(HoUi) = |\{Ni, Ui\}| = 2 \) in the case of Figure 5.2.

\[ \forall e \in E : \mu_s(e) = \delta_{sbt}(\alpha(e), \tau(e)) \text{ measures } a \text{ of } e, \text{ where} \]
\[ \mu, \alpha, \tau, \text{ and } a \text{ are tabulated in Table 5.24} \]

Table 5.23: Behavioural Coupling Measures Definition (Scenario Perspective)

### 5.4.4 Indirect coupling

In this section, behavioural coupling sub-attributes are measured taking into account indirect interactions—which have not been considered in previous sections—between components of a DIS behavioural model. Indirect coupling is measured by the length of scenario interaction chains. Indirect coupling may be crucial in the comparison of components with the same or similar amount of direct coupling.

It should be noted that:

- indirect coupling is only considered from an outbound viewpoint, since we do not believe inbound indirect coupling has a significant impact on the quality attributes of interest; and
Table 5.24: Behavioural Coupling Measures Legend (Scenario Perspective)

- indirect coupling cannot be measured from a scenario or component perspective, only from the perspective of interactions.

Therefore, we consider only 4 coupling sub-attributes, namely: synchronisation (SL), availability (AL), conversation (CL), and static binding (BL) indirect coupling.

5.4.4.1 Measurement abstractions

Each of these sub-attributes requires an appropriate measurement abstraction to be defined. The abstraction function \( \alpha \) for a specific coupling sub-attribute returns, given a distributed component \( c \), the components of the longest path of interaction of which \( c \) is part. A path is, hence, a set of components. The measurement abstraction functions are defined in Table 5.25.

In the example of Figure 5.2, \( \alpha_{SL}(SMa) = \{SBs, SDs\} \) and \( \alpha_{SL}(WbBs) = \{WbDs\} \).

5.4.4.2 Elementary transformations and metric space

The elements of \( M \) are also sets (of components). Therefore, the generic metric space for sets defined above in Section 5.2.1 on page 84 can be used too.

5.4.4.3 Reference abstractions

There might be components that are not coupled to any other components in a DIS behavioural model. Therefore empty sets are chosen as reference abstractions for
CHAPTER 5. MEASURE DEFINITION AND THEORETICAL VALIDATION

Let $E$ be the set of components associated with some Universe of Discourse (i.e., a behavioural model $B$). Then $\forall e \in E$:

$\alpha_{SL}(e) = \{f \in E | f$ is part of the longest path of synchronous interactions of $e\}$

$\alpha_{AL}(e) = \{f \in E | f$ is part of the longest path of available interactions of $e\}$

$\alpha_{CL}(e) = \{f \in E | f$ is part of the longest path of conversational interactions of $e\}$

$\alpha_{BL}(e) = \{f \in E | f$ is part of the longest path of static bound interactions of $e\}$

Table 5.25: Behavioural Coupling Measurement Abstractions (Indirect Perspective)

SL, AL, CL, and BL coupling. Formally,

$\forall e \in E : \tau_{SL}(e) = \tau_{AL}(e) = \tau_{CL}(e) = \tau_{BL}(e) = \emptyset$

Table 5.26: Behavioural Coupling Reference Abstractions (Indirect Perspective)

5.4.4.4 Measures Definition

To avoid repetition, we present a generic software measure definition in Table 5.27. Given the choices regarding measurement abstractions, reference abstractions, and metric made above, the behavioural indirect coupling measures return for a component $e$ the count of components which are part of the longest path of interactions (of the relevant type) in the DIS behavioural model. In other words, the measure captures the conceptual distance between the measurement abstraction and the reference abstraction. Note indirect interactions between components are taken into account by all the abstractions.

$\forall e \in E : \mu_{k}(e) = \delta_{SET}(\alpha(e), \tau(e))$ measures $a$ of $e$, where

$\mu, \alpha, \tau$, and $a$ are tabulated in Table 5.28

Table 5.27: Behavioural Coupling Measures Definition (Indirect Perspective)
### 5.5. SUMMARY

<table>
<thead>
<tr>
<th>Attribute (α)</th>
<th>Measurement Abstraction (α)</th>
<th>Reference Abstraction (τ)</th>
<th>Measure (μ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SL coupling</td>
<td>$\alpha_{SL}$</td>
<td>$\tau_{SL}$</td>
<td>SL</td>
</tr>
<tr>
<td>AL coupling</td>
<td>$\alpha_{AL}$</td>
<td>$\tau_{AL}$</td>
<td>AL</td>
</tr>
<tr>
<td>CL coupling</td>
<td>$\alpha_{CL}$</td>
<td>$\tau_{CL}$</td>
<td>CL</td>
</tr>
<tr>
<td>BL coupling</td>
<td>$\alpha_{BL}$</td>
<td>$\tau_{BL}$</td>
<td>BL</td>
</tr>
</tbody>
</table>

Table 5.28: Behavioural Coupling Measures Legend (Indirect Perspective)

#### 5.5 Summary

This chapter has presented the definition of the proposed structural and behavioural measures following a formal approach. A summary of all proposed measures can be found in Table 5.29. In addition, Table 5.30 presents a comparison of these measures with existing proposals.

The theoretical validity of the measures (according to the Measurement Theory prescriptions) is guaranteed by the approach followed. The following chapter presents an empirical validation of the measures as useful estimators of quality attributes of practical interest.
<table>
<thead>
<tr>
<th>Structural</th>
<th>Behavioural</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inbound Coupling</td>
<td></td>
</tr>
<tr>
<td>CLIP</td>
<td>CIS</td>
</tr>
<tr>
<td>CLI</td>
<td>CIA</td>
</tr>
<tr>
<td>COIP</td>
<td>CIC</td>
</tr>
<tr>
<td>COII</td>
<td>CIB</td>
</tr>
<tr>
<td>DIP</td>
<td>SIS</td>
</tr>
<tr>
<td>DII</td>
<td>SIA</td>
</tr>
<tr>
<td>Outbound Coupling</td>
<td></td>
</tr>
<tr>
<td>CLOP</td>
<td>COS</td>
</tr>
<tr>
<td>CLOI</td>
<td>COA</td>
</tr>
<tr>
<td>COOP</td>
<td>COC</td>
</tr>
<tr>
<td>COOI</td>
<td>COB</td>
</tr>
<tr>
<td>DOP</td>
<td>SOS</td>
</tr>
<tr>
<td>DOII</td>
<td>SOA</td>
</tr>
<tr>
<td>SOII</td>
<td>SOC</td>
</tr>
<tr>
<td>SOOB</td>
<td>SOB</td>
</tr>
</tbody>
</table>

Table 5.29: Summary of the proposed measures

<table>
<thead>
<tr>
<th>Source</th>
<th>Attribute</th>
<th>Artefact</th>
<th>T.E.</th>
<th>E.E.</th>
<th>T.I.</th>
<th>Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shatz [1988]</td>
<td>Complexity</td>
<td>Code</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Cheng [1993]</td>
<td>Complexity</td>
<td>Code</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Tsaur and Horng [1998]</td>
<td>Complexity</td>
<td>Code</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Morasca [1999]</td>
<td>Size, Length, Complexity, Coupling</td>
<td>Specifications</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td><strong>This thesis</strong></td>
<td><strong>Coupling</strong></td>
<td><strong>Design</strong></td>
<td><strong>Yes</strong></td>
<td><strong>Yes</strong></td>
<td><strong>Yes</strong></td>
<td><strong>Yes</strong></td>
</tr>
</tbody>
</table>

Chapter 6

Empirical Evaluation – Structural Measures

6.1 Introduction

As we have stated previously, theoretical validation by itself is not enough. The next step is to carry out the empirical evaluation of the proposed measures against the quality attributes of interest, namely efficiency, reliability and maintainability. Therefore, in this chapter the empirical validation of the structural measures is presented. Three experiments and two replications were carried out, which are reported following the guidelines outlined by Wohlin et al. [2000].

6.2 Definition

The experimental goals were defined as shown in Table 6.1 following the GQM template [Basili and Rombach, 1988].

6.3 Planning

6.3.1 Context

The DIS studied is a proof-of-concept enterprise DIS that supports the academic management of an Australian university. The system, developed by a team of grad-
|   | Analyse structural coupling measures,  
|   | for the purpose of evaluation,  
|   | with respect to their capability of being used as indicators of efficiency,  
|   | from the point of view of DIS software engineers,  
|   | in the context of a proof-of-concept DIS developed by a team of graduates.  
|   | Analyse structural coupling measures,  
|   | for the purpose of evaluation,  
|   | with respect to their capability of being used as indicators of reliability,  
|   | from the point of view of DIS software engineers,  
|   | in the context of a proof-of-concept DIS developed by a team of graduates.  
|   | Analyse structural coupling measures,  
|   | for the purpose of evaluation,  
|   | with respect to their capability of being used as indicators of maintainability,  
|   | from the point of view of DIS software engineers,  
|   | in the context of a proof-of-concept DIS developed by a team of graduates.  
|   | Table 6.1: Experimental Goals  

uates, is entirely written in Java and it is composed of 21 distributed components,  
67 classes and about 11,000 lines of code. The middleware used by components to  
interact are JDBC [Sun, 2003a], JRMI [Sun, 2003c] and JMS [Sun, 2003b].

### 6.3.2 Experiment Design

In order to evaluate software measurement hypotheses empirically, it is possible to  
adopt two main strategies [Briand et al., 1999]:

- small-scale controlled experiments, and/or
- real-scale industrial case studies.

In this case, for an exploratory investigation, we chose the first alternative since it is more suitable to study the phenomena of interest in isolation, without having to deal with other sources of variation, such as co-existing systems and security mechanisms. However, we envisage that after several experiments the suite of measures will be shown to be robust, and we intend to test the measures utilising the second strategy—it should be noted that this is outside the scope of this thesis.
6.3.3 Hypotheses

The experimental hypotheses 1, 2 and 3 established in section 4.3.2 are redefined here by refining the design attribute:

- **Hypothesis 1.1.** The structural OP coupling of a distributed cluster affects its extrinsic efficiency.

- **Hypothesis 1.2.** The structural OI coupling of a distributed cluster affects its extrinsic efficiency.

- **Hypothesis 1.3.** The structural IP coupling of a distributed cluster affects its extrinsic efficiency.

- **Hypothesis 1.4.** The structural II coupling of a distributed cluster affects its extrinsic efficiency.

- **Hypothesis 2.1.** The extrinsic reliability of a distributed cluster is affected by its structural OP coupling.

- **Hypothesis 2.2.** The extrinsic reliability of a distributed cluster is affected by its structural OI coupling.

- **Hypothesis 2.3.** The extrinsic reliability of a distributed cluster is affected by its structural IP coupling.

- **Hypothesis 2.4.** The extrinsic reliability of a distributed cluster is affected by its structural II coupling.

- **Hypothesis 3.1.** The structural OP coupling of a distributed cluster affects its extrinsic maintainability.

- **Hypothesis 3.2.** The structural OI coupling of a distributed cluster affects its extrinsic maintainability.

- **Hypothesis 3.3.** The structural IP coupling of a distributed cluster affects its extrinsic maintainability.

- **Hypothesis 3.4.** The structural II coupling of a distributed cluster affects its extrinsic maintainability.
6.3.4 Variables

Structural coupling is quantified by the independent measures defined in Chapter 5 and shown in Table 6.2.

As discussed previously, efficiency, reliability and maintainability are high-level quality attributes that are evaluated by targeting one of its sub-attributes. Table 6.3 shows the targeted quality sub-attributes, their definitions (according to ISO 9126-1) and the specific dependent measures selected to quantify them. The dependent measures can be described as follows:

- The times $t_b$ and $t_a$ are measured immediately before and after each remote interaction, and the difference $(t_a - t_b)$ is considered the blocking time of the interaction. TBT is the sum of all blocking times due to remote interactions.

- The occurrence of a failure is recorded whenever a remote interaction throws an exception. TNF is the sum of all failures due to remote interactions.

- OFC is calculated as the ratio of LET/TET, where Local Execution Time ($LET$) is the difference between Total Execution Time ($TET$) and Remote Execution Time ($RET$), that is $LET = TET - RET$. $RET$ is the time spent satisfying remote interactions. The higher the $RET$, the lower the OFC, since a cluster cannot be disconnected from the rest of the system for maintenance while it is interacting with remote clusters.

6.3.5 Instrumentation

A software tool was developed to extract generic high-level design information from the structure of the system. This information was consequently used to compute
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Sub-attribute</th>
<th>Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Efficiency</td>
<td><em>Time Behaviour:</em> the capability of the software product to provide appropriate performance and processing times and throughput rates when performing its function, under stated conditions</td>
<td><em>TBT:</em> Total Blocking Time</td>
</tr>
<tr>
<td>2 Reliability</td>
<td><em>Maturity:</em> the capability of the software product to avoid failure as a result of faults in the system</td>
<td><em>TNF:</em> Total Number of Failures</td>
</tr>
<tr>
<td>3 Maintainability</td>
<td><em>Changeability:</em> the capability of the software product to enable a specified modification to be implemented</td>
<td><em>OFC:</em> Opportunity For Change</td>
</tr>
</tbody>
</table>

Table 6.3: Quality Measures

the measures presented above—and eventually others that might be defined. Monitoring code was added to the implementation of the system in appropriate locations to obtain the measurement data of the quality attributes.

### 6.4 Operation

#### 6.4.1 Preparation

In addition to the monitoring code, special code was introduced to some components to simulate the random operation of the system by end-users. Before the actual experiments, several pilot experiments were run to make sure that there were no apparent anomalies, and that the system behaved in the same way after the monitoring and simulation code was introduced.

In the second experiment, extra code was also added to simulate the occurrence of random faults in the system.

#### 6.4.2 Execution

The experiments were conducted in the Distributed Computing Laboratory of RMIT University. The system was executed on an isolated network of workstations, each
of which equipped with a single CPU and running under the Linux operating system. All computers had the same hardware and software, and were configured in the same way—the same binary image was used for all hard disks. Every component was run on a separate workstation as the only user process, all other processes running were a few system processes started by default. The execution of the system was initiated and terminated by the experiment team, which also controlled that in the meantime nobody else had access to the facilities.

### 6.4.3 Data validation

Despite the data being collected reliably and objectively by electronic means, it was thoroughly inspected to assert that it was consistent. For this purpose we ran each experiment at five different times and compared the observed data. There was no need to discard any data, thus all data collected was used. However, since the results were very similar (difference < 2.5%) for the each experiment, only three data sets per experiment are reported to avoid repetition. All combinations produced the same results, therefore we simply omitted the data sets of the last two runs in all cases.

### 6.5 Analysis and Interpretation of the Results

After the execution of the experiments, all measures were computed from the recorded data. The empirical data was analysed with the assistance of the software package SPSS [1998]. Raw experimental data can be found in Appendix A.

#### 6.5.1 Descriptive statistics

Tables 6.4 and 6.5 present the minimum, maximum, sum, mean, standard deviation and variance of the structural measures and the quality measures. The histogram plots of all measures suggested that their distributions were not normal. This was confirmed by the Shapiro-Wilk test. The results show clearly that the distribution of the measures deviated significantly from normality—see Tables 6.6 and 6.7.
6.5. ANALYSIS AND INTERPRETATION OF THE RESULTS

<table>
<thead>
<tr>
<th>Measure</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Sum</th>
<th>Mean</th>
<th>Std. Dev.</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLOP</td>
<td>0.00</td>
<td>3.00</td>
<td>30.00</td>
<td>1.43</td>
<td>1.08</td>
<td>1.16</td>
</tr>
<tr>
<td>CLOI</td>
<td>0.00</td>
<td>2.00</td>
<td>5.00</td>
<td>0.24</td>
<td>0.62</td>
<td>0.39</td>
</tr>
<tr>
<td>CLIP</td>
<td>0.00</td>
<td>5.00</td>
<td>30.00</td>
<td>1.43</td>
<td>1.63</td>
<td>2.65</td>
</tr>
<tr>
<td>CLII</td>
<td>0.00</td>
<td>2.00</td>
<td>5.00</td>
<td>0.24</td>
<td>0.62</td>
<td>0.39</td>
</tr>
<tr>
<td>COOP</td>
<td>0.00</td>
<td>1.00</td>
<td>15.00</td>
<td>0.71</td>
<td>0.46</td>
<td>0.21</td>
</tr>
<tr>
<td>COOI</td>
<td>0.00</td>
<td>1.00</td>
<td>3.00</td>
<td>0.14</td>
<td>0.36</td>
<td>0.13</td>
</tr>
<tr>
<td>COIP</td>
<td>0.00</td>
<td>1.00</td>
<td>15.00</td>
<td>0.71</td>
<td>0.46</td>
<td>0.13</td>
</tr>
<tr>
<td>COII</td>
<td>0.00</td>
<td>1.00</td>
<td>3.00</td>
<td>0.14</td>
<td>0.36</td>
<td>0.13</td>
</tr>
<tr>
<td>DOP</td>
<td>0.00</td>
<td>3.00</td>
<td>30</td>
<td>1.43</td>
<td>1.08</td>
<td>1.16</td>
</tr>
<tr>
<td>DOI</td>
<td>0.00</td>
<td>2.00</td>
<td>5.00</td>
<td>0.24</td>
<td>0.62</td>
<td>0.39</td>
</tr>
<tr>
<td>DIP</td>
<td>0.00</td>
<td>5.00</td>
<td>30.00</td>
<td>1.43</td>
<td>1.63</td>
<td>2.65</td>
</tr>
<tr>
<td>DII</td>
<td>0.00</td>
<td>3.00</td>
<td>5.00</td>
<td>0.24</td>
<td>0.62</td>
<td>0.39</td>
</tr>
</tbody>
</table>

Table 6.4: Structural Measures: Descriptive Statistics \( (n = 21) \)

<table>
<thead>
<tr>
<th>Measure</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Sum</th>
<th>Mean</th>
<th>Std. Dev.</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBT_1</td>
<td>0.00</td>
<td>11.57</td>
<td>85.90</td>
<td>4.09</td>
<td>4.14</td>
<td>17.14</td>
</tr>
<tr>
<td>TBT_2</td>
<td>0.00</td>
<td>11.59</td>
<td>84.51</td>
<td>4.02</td>
<td>4.07</td>
<td>16.60</td>
</tr>
<tr>
<td>TBT_3</td>
<td>0.00</td>
<td>11.83</td>
<td>86.46</td>
<td>4.12</td>
<td>4.18</td>
<td>17.45</td>
</tr>
<tr>
<td>TNF_1</td>
<td>0.00</td>
<td>12.44</td>
<td>88.67</td>
<td>4.22</td>
<td>4.61</td>
<td>21.22</td>
</tr>
<tr>
<td>TNF_2</td>
<td>0.00</td>
<td>12.11</td>
<td>88.84</td>
<td>4.23</td>
<td>4.54</td>
<td>20.57</td>
</tr>
<tr>
<td>TNF_3</td>
<td>0.00</td>
<td>12.31</td>
<td>92.30</td>
<td>4.39</td>
<td>4.67</td>
<td>21.79</td>
</tr>
<tr>
<td>OFC_1</td>
<td>31.93</td>
<td>100.00</td>
<td>1720.40</td>
<td>81.92</td>
<td>23.56</td>
<td>554.91</td>
</tr>
<tr>
<td>OFC_2</td>
<td>35.27</td>
<td>100.00</td>
<td>1725.87</td>
<td>82.18</td>
<td>23.03</td>
<td>530.44</td>
</tr>
<tr>
<td>OFC_3</td>
<td>35.27</td>
<td>100.00</td>
<td>1725.87</td>
<td>82.18</td>
<td>23.03</td>
<td>530.44</td>
</tr>
</tbody>
</table>

Table 6.5: Quality Measures: Descriptive Statistics \( (n = 21) \)
<table>
<thead>
<tr>
<th>Measure</th>
<th>Shapiro-Wilk Statistic</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLOP</td>
<td>0.840</td>
<td>0.003</td>
</tr>
<tr>
<td>CLOI</td>
<td>0.428</td>
<td>0.000</td>
</tr>
<tr>
<td>CLIP</td>
<td>0.736</td>
<td>0.000</td>
</tr>
<tr>
<td>CLIi</td>
<td>0.428</td>
<td>0.000</td>
</tr>
<tr>
<td>COOP</td>
<td>0.570</td>
<td>0.000</td>
</tr>
<tr>
<td>COOI</td>
<td>0.422</td>
<td>0.000</td>
</tr>
<tr>
<td>COIP</td>
<td>0.570</td>
<td>0.000</td>
</tr>
<tr>
<td>COII</td>
<td>0.422</td>
<td>0.000</td>
</tr>
<tr>
<td>DOP</td>
<td>0.840</td>
<td>0.003</td>
</tr>
<tr>
<td>DOI</td>
<td>0.428</td>
<td>0.000</td>
</tr>
<tr>
<td>DIP</td>
<td>0.736</td>
<td>0.000</td>
</tr>
<tr>
<td>DII</td>
<td>0.428</td>
<td>0.000</td>
</tr>
</tbody>
</table>

Table 6.6: Structural Measures: Normality Test \((n = 21)\)

<table>
<thead>
<tr>
<th>Measure</th>
<th>Shapiro-Wilk Statistic</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 TBT_1</td>
<td>0.859</td>
<td>0.006</td>
</tr>
<tr>
<td>TBT_2</td>
<td>0.861</td>
<td>0.007</td>
</tr>
<tr>
<td>TBT_3</td>
<td>0.861</td>
<td>0.007</td>
</tr>
<tr>
<td>2 TNF_1</td>
<td>0.819</td>
<td>0.001</td>
</tr>
<tr>
<td>TNF_2</td>
<td>0.825</td>
<td>0.002</td>
</tr>
<tr>
<td>TNF_3</td>
<td>0.821</td>
<td>0.001</td>
</tr>
<tr>
<td>3 OFC_1</td>
<td>0.766</td>
<td>0.000</td>
</tr>
<tr>
<td>OFC_2</td>
<td>0.769</td>
<td>0.000</td>
</tr>
<tr>
<td>OFC_3</td>
<td>0.769</td>
<td>0.000</td>
</tr>
</tbody>
</table>

Table 6.7: Quality Measures: Normality Test \((n = 21)\)
All structural measures are simple counts. Hence, their values are non-negative integer numbers and their minimum value is zero. All structural measures were defined and theoretically validated on the ratio scale in Chapter 5. The observations corresponding to the measures COOI, COII, CLOI, CLI, DOI and DII had most of their values equal to zero. Therefore they were excluded from the correlation analysis—this approach is also followed by El Emam et al. [2001].

TBT is a time measure and its units are hundreds of seconds; TNF is a non-negative integer and its units are hundreds of failures; and OFC is a percentage, hence a non-negative real number less than or equal to 100. It is important to remark that the different measurements of TBT, TNF and OFC are highly consistent. That is, it can be seen that similar data is observed from each run of each experiment.

6.5.2 Correlation Analysis

As the data from all the measures was not normally distributed, Spearman, a non-parametric statistic was used. Non-significant correlation coefficients are not shown, they have been substituted by asterisks (*). These measures will not be considered in the regression analysis.

6.5.2.1 Efficiency

Table 6.8 presents the correlation coefficients (at the 0.01 significance level) between the structural measures and efficiency—measured by TBT. The results show that the correlations of the measures of OP coupling (namely COOP, CLOP and DOP) with TBT are statistically significant. Furthermore, these correlation coefficients are highly significant, indicating a nontrivial association of these measures with efficiency—measured by TBT. This also suggests that these variables are candidates for a base regression model to estimate efficiency in terms of TBT. Examination of the correlation coefficients suggests that these three measures are (positively) correlated to TBT. It should be noted that a higher value of TBT indicates worse efficiency.

Therefore, the hypothesis (1.1) underlying the measures of OP coupling seems to
Table 6.8: Structural measures and TBT: Correlation coefficients

be supported in our environment. Clusters with higher OP coupling exhibit higher values of TBT, and hence worse efficiency. On the other hand, the correlations of the measures of IP coupling (namely COIP, CLIP and DIP) are not statistically significant. Hence, hypothesis 1.3 is not supported in our context. Due to unsuitability of the available data of the OI and II coupling measures, we were unable to test hypotheses 1.2 and 1.4.

These findings cannot be considered final, they can only be considered preliminary. Further experimentation is indispensable to extract conclusive results. It is necessary to replicate the experiment and to carry out new ones to confirm the tested hypotheses and to evaluate the untested hypotheses.

6.5.2.2 Reliability

The correlation coefficients (at the 0.01 significance level) between the structural measures and reliability, measured by TNF, are presented in Table 6.9. The results show that the correlations of the measures of OP coupling (namely COOP, CLOP and DOP) with TNF are statistically significant. Furthermore, these correlation coefficients are highly significant, indicating a nontrivial association of these measures with reliability—measured by TNF. This also suggests that these variables are candidates for a base regression model to estimate reliability in terms of TNF. Examination of the correlation coefficients suggests that these three measures are (positively) correlated to TNF. It should be noted that a higher value of TNF indicates worse reliability.

Consequently, the hypothesis (2.1) underlying the measures of OP coupling
### 6.5. ANALYSIS AND INTERPRETATION OF THE RESULTS

<table>
<thead>
<tr>
<th></th>
<th>TNF₁</th>
<th>TNF₂</th>
<th>TNF₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLOP</td>
<td>0.88</td>
<td>0.89</td>
<td>0.89</td>
</tr>
<tr>
<td>CLIP</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>COOP</td>
<td>0.65</td>
<td>0.65</td>
<td>0.65</td>
</tr>
<tr>
<td>COIP</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>DOP</td>
<td>0.88</td>
<td>0.89</td>
<td>0.89</td>
</tr>
<tr>
<td>DIP</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

Table 6.9: Structural measures and TNF: Correlation coefficients

seems to be supported in our environment. Clusters with higher OP coupling exhibit higher values of TNF, and hence worse reliability. On the other hand, the correlations of the measures of IP coupling (namely COIP, CLIP and DIP) are not statistically significant. Therefore, hypothesis 2.3 is not supported in our context. Due to unsuitability of the available data of the OI and II coupling measures, we were unable to test hypotheses 2.2 and 2.4.

These findings cannot be considered final, they can only be considered preliminary. Further experimentation is indispensable to extract conclusive results. It is necessary to replicate the experiment and to carry out new ones to confirm the tested hypotheses and to evaluate the untested hypotheses.

#### 6.5.2.3 Maintainability

Table 6.10 presents the Spearman’s correlation coefficients (at the 0.01 significance level) between the structural measures and maintainability—measured by OFC. The results show that the correlations of the measures of IP coupling (namely COIP, CLIP and DIP) with OFC are statistically significant. Furthermore, these correlation coefficients are highly significant, indicating a nontrivial association of these measures with maintainability—measured by OFC. This also suggests that these variables are candidates for a base regression model to estimate maintainability in terms of OFC. Examination of the correlation coefficients suggests that these three measures are (negatively) correlated to OFC.

Therefore, the hypothesis (3.3) underlying the measures of IP coupling seems to be supported in our environment. Clusters with higher IP coupling exhibit lower
values of OFC, and hence worse maintainability. On the other hand, the correlations of the measures of OP coupling (namely COOP, CLOP and DOP) are not statistically significant. Hence, hypothesis 3.1 is not supported in our context. Due to unsuitability of the available data of the OI and II coupling measures, we were unable to test hypotheses 3.2 and 3.4.

These findings cannot be considered final, they can only be considered preliminary. Further experimentation is indispensable to extract conclusive results. It is necessary to replicate the experiment and to carry out new ones to confirm the tested hypotheses and to evaluate the untested hypotheses.

### 6.5.3 Simple Regression Analysis

In this section, we present the results obtained when analysing the individual impact of the structural measures on quality attributes using regression analysis. As discussed in Chapter 3, a multiple linear regression equation has the following form:

$$ Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n + \varepsilon \quad (6.1) $$

where $Y$ is the response variable, $X_i$ are the explanatory variables, and $\beta_i$ are the regression coefficients. A simple regression model is a special case of this, where only one explanatory variable appears.

Table 6.11 presents the computed regression coefficients ($\hat{\beta}$), the actual statistical significance of $\beta$, and indicators ($R$, $R^2$ and adjusted $R^2$) of goodness-of-fit
of the efficiency models. Each row contains the parameter estimates of a simple regression model.

The results obtained indicate that the measures considered in this section (namely CLOP, COOP and DOP) not only correlate with efficiency, but are also useful to explain the variation in TBT. For instance, in our environment, OP coupling (measured by CLOP) accounted for 77 percent of the variation in efficiency (measured by TBT), and each increase of one unit of CLOP increased the TBT by 337 seconds.

The computed regression coefficients ($\beta$), the actual statistical significance of $\beta$, and indicators ($R$, $R^2$ and adjusted $R^2$) of goodness-of-fit of the reliability models are presented in Table 6.12. Each row contains the parameter estimates of a simple regression model.

The results obtained indicate that the measures considered in this section (namely CLOP, COOP and DOP) not only correlate with reliability, but are also useful to explain the variation in TNF. For example, in our environment, OP coupling (measured by DOP) accounted for 73 percent of the variation in reliability (measured by TNF), and each increase of one unit of DOP increased the TNF by 360 failures.

Table 6.13 presents the computed regression coefficients ($\beta$), the actual statistical significance of $\beta$, and indicators ($R$, $R^2$ and adjusted $R^2$) of goodness-of-fit of
the maintainability models. Each row contains the parameter estimates of a simple regression model.

The results obtained indicate that only COIP is useful to explain the variation in OFC. In our environment, IP coupling (measured by COIP) accounted for 25 percent of the variation in maintainability (measured by OFC), and each increase of one unit of COIP decreased the OFC by 25 units.
6.5.4 Multiple Regression Analysis

It is important to note that we do not expect the structural measures to explain all the variation of quality attributes, since other factors are likely to be important too, such as network bandwidth, system load, computer and hardware robustness. However, the purpose of multiple regression analysis is to determine whether the measures appearing significant in the simple regression analysis are complementary.

Furthermore, in this dissertation we are interested in the goodness-of-fit of the models, and did not investigate the predictive capability of the models per se. However, a satisfactory goodness of fit is necessary in order to expect a satisfactory predictive capability in future studies.

The analysis of residuals is a simple yet powerful tool for evaluating the appropriateness of the multiple regression models [Freund and Wilson, 1998]. Consequently, the underlying assumptions will be evaluated:

- **Homogeneity of the error term variance.** The variance of the error terms should not be very unequal.

- **Independence of the error term.** The error terms should not be correlated.

- **Normality of the error term distribution.** The errors should be random and normally distributed.

6.5.4.1 Efficiency

Table 6.14 provides the estimated regression coefficients ($\beta$) and their significance ($\rho$) based on a $t$ test, for the structural measures model after performing stepwise multiple linear regression [Freund and Wilson, 1998]. Multiple regression models show a better goodness-of-fit than simple regression models, so the measures (in combination) are deemed to be potentially useful for building a model to predict efficiency. The fact that the significance for $\beta_0$ is greater than 0.05 only means that we cannot conclude that $\beta_0 \neq 0$. However, we can conclude that it is very unlikely that $\beta_i = 0$ with $i = 1, 2$. Thus, it is very likely that COOP and CLOP are correlated to efficiency.
The underlying assumptions have been evaluated as follows:

- **Homogeneity of the error term variance.** This assumption holds since the plots of standardised residuals against the standardised predicted values (e.g. Figure 6.1) show a random scatter of points with no discernible pattern identified.

- **Independence of the error term.** We tested this assumption with a Durbin-Watson test that did not reveal a significant correlation between residuals (see Table 6.15). This statistic can vary between 0 and 4 with values between 1.5 and 2.5 indicating that residuals are not correlated strongly.

- **Normality of the error term distribution.** The results of the Shapiro-Wilk test are not significant when $\alpha = 0.05$, therefore we can conclude that this assumption is tenable (see Table 6.16).

Finally, DOP is not included in the models because it is highly correlated to CLOP. Therefore, since the inclusion of DOP to the model does not increase $R$, this variable in considered redundant.
6.5. ANALYSIS AND INTERPRETATION OF THE RESULTS

Figure 6.1: Scatter plot of residuals versus predicted values (TBT₂)

<table>
<thead>
<tr>
<th></th>
<th>$Y = TBT_1$</th>
<th>$Y = TBT_2$</th>
<th>$Y = TBT_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shapiro-Wilk Statistic</td>
<td>0.91</td>
<td>0.92</td>
<td>0.91</td>
</tr>
<tr>
<td>Significance</td>
<td>0.057</td>
<td>0.092</td>
<td>0.053</td>
</tr>
</tbody>
</table>

Table 6.16: Normality of the residuals (TBT)
<table>
<thead>
<tr>
<th></th>
<th>$X_i$</th>
<th>$\beta_i$</th>
<th>Sig.</th>
<th>$Y$</th>
<th>$R$</th>
<th>$R^2$</th>
<th>Adj. $R^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>constant</td>
<td>0.00</td>
<td>&gt; 0.05</td>
<td>$\text{TNF}_1$</td>
<td>0.88</td>
<td>0.77</td>
<td>0.75</td>
<td></td>
</tr>
<tr>
<td>DOP</td>
<td>5.43</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COOP</td>
<td>-4.96</td>
<td>0.04</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>constant</td>
<td>0.00</td>
<td>&gt; 0.05</td>
<td>$\text{TNF}_2$</td>
<td>0.89</td>
<td>0.80</td>
<td>0.77</td>
<td></td>
</tr>
<tr>
<td>DOP</td>
<td>5.44</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COOP</td>
<td>-4.95</td>
<td>0.03</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>constant</td>
<td>0.00</td>
<td>&gt; 0.05</td>
<td>$\text{TNF}_3$</td>
<td>0.90</td>
<td>0.81</td>
<td>0.79</td>
<td></td>
</tr>
<tr>
<td>DOP</td>
<td>5.65</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COOP</td>
<td>-5.14</td>
<td>0.02</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.17: Structural Measures and TNF: Multiple Regression Statistics

### 6.5.4.2 Reliability

Table 6.17 provides the estimated regression coefficients ($\beta$) and their significance ($\rho$) based on a $t$ test, for the structural measures model after performing stepwise multiple linear regression [Freund and Wilson, 1998]. Multiple regression models show a better goodness-of-fit than simple regression models, so the measures (in combination) are deemed to be potentially useful for building a model to predict reliability. The fact that the significance for $\beta_0$ is greater than 0.05 only means that we cannot conclude that $\beta_0 \neq 0$. However, we can conclude that it is very unlikely that $\beta_i = 0$ with $i = 1, 2$. Thus, it is very likely that COOP and DOP are correlated to reliability.

The underlying assumptions have been evaluated as follows:

- **Homogeneity of the error term variance.** This assumption holds since the plots of standardised residuals against the standardised predicted values (e.g. Figure 6.2) show a random scatter of points with no discernible pattern identified.

- **Independence of the error term.** We tested this assumption with a Durbin-Watson test that did not reveal a significant correlation between residuals (see Table 6.18). This statistic can vary between 0 and 4 with values between 1.5 and 2.5 indicating that residuals are not correlated strongly.

- **Normality of the error term distribution.** The results of the Shapiro-Wilk test are
not significant when $\alpha = 0.05$, therefore we can conclude that this assumption is tenable (see Table 6.19).

Finally, CLOP is not included in the models because it is highly correlated to DOP. Therefore, since the inclusion of CLOP to the model does not increase $R$, this variable is considered redundant.

<table>
<thead>
<tr>
<th></th>
<th>$Y = TFN_1$</th>
<th>$Y = TFN_2$</th>
<th>$Y = T N F_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Durbin-Watson Statistic</td>
<td>1.756</td>
<td>1.754</td>
<td>1.704</td>
</tr>
</tbody>
</table>

Table 6.18: Independence of the residuals (TNF)
\centering
\begin{tabular}{|l|c|c|c|}
\hline
 & $Y = TNF_1$ & $Y = TNF_2$ & $Y = TNF_3$ \\
\hline
Shapiro-Wilk Statistic & 0.751 & 0.760 & 0.783 \\
Significance & 0.000 & 0.000 & 0.000 \\
\hline
\end{tabular}
\caption{Normality of the residuals (TNF)}

6.5.4.3 **Maintainability**

Since only one structural measure appears to be a good predictor of OFC, multiple models are not possible.

### 6.6 Threats

Four different threats to the validity of the experiments should be addressed [Wohlin et al., 2000]:

- **Conclusion validity.** An issue that could affect the statistical validity of these studies is the size of the sample data, which may not be large enough for a conclusive statistical analysis. Thus, we do not consider these results to be final.

- **Construct validity.** The studies were carefully designed, and they were piloted several times before actually being run. The quality measures are times (TBT), simple counts (TNF) and percentages (OFC), which can be objectively and reliably measured. The structural measures used in this study were shown to adequately quantify the attribute they purport to measure in Chapter 5.

- **Internal validity.** The experiments were highly controlled and monitored, so it is very unlikely that undetected influences have occurred without our knowledge. The consistency among observed data sets obtained independently also supports this. The instrumentation was trustworthy since the data was collected, and the measures computed, electronically.

- **External validity.** Although the studies are based on a real case, more studies are needed using systems from different enterprise domains. We are aware
that more experiments with different platforms (e.g. computer and network hardware, operating systems, etc.) and infrastructure (e.g. middleware type, programming language) must also be carried out to further generalise these results. As a first step in this direction, we replicated the experiments twice with different platforms, the results of which are presented in section 6.7.

6.7 Replication of Experiments

As a first step to generalise the results obtained in the previous sections, we replicated these experiments running the system over different platforms. In this section, we report the analysis of the results only, since their definition, planning and operation of the experiments remain identical, except for the following:

- The platform of the computer network was different: the Linux operating system was changed to the Windows operating system in one case, and in the other case the network Ethernet links were replaced by wireless links.

- Each replication was run three (instead of five) different times to check the consistency of the data, and only one (instead of three) data set is reported for the same reason—i.e. to avoid repetition.

6.7.1 First Experiment Replications

6.7.1.1 Descriptive statistics

Table 6.20 presents the descriptive statistics of TBT in bold. A comparison with the original experiment shows that they are similar in the case of the first replication, while they exhibit higher values in the case of the second replication, in principle due to the impact of a slower network. In spite of this, the distribution of the data is similar to the original experiment.

The histogram plots suggested that their distributions were not normal either. This was confirmed by the Shapiro-Wilk test. The results show clearly that the distributions also deviated significantly from normality (see Table 6.21).
CHAPTER 6. EMPirical evaluation – Structural Measures

<table>
<thead>
<tr>
<th>Measure</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Sum</th>
<th>Mean</th>
<th>Std. Dev.</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBT₄</td>
<td>0.00</td>
<td>12.15</td>
<td>87.63</td>
<td>4.17</td>
<td>4.31</td>
<td>18.54</td>
</tr>
<tr>
<td>TBT₅</td>
<td>0.00</td>
<td>16.55</td>
<td>118.37</td>
<td>5.63</td>
<td>5.91</td>
<td>34.92</td>
</tr>
<tr>
<td>TBT₂</td>
<td>0.00</td>
<td>11.59</td>
<td>84.51</td>
<td>4.02</td>
<td>4.07</td>
<td>16.60</td>
</tr>
</tbody>
</table>

Table 6.20: TBT: Descriptive Statistics (n = 21)

<table>
<thead>
<tr>
<th>Measure</th>
<th>Shapiro-Wilk Statistic</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBT₄</td>
<td>0.852</td>
<td>0.005</td>
</tr>
<tr>
<td>TBT₅</td>
<td>0.848</td>
<td>0.004</td>
</tr>
<tr>
<td>TBT₂</td>
<td>0.861</td>
<td>0.007</td>
</tr>
</tbody>
</table>

Table 6.21: TBT: Normality Test (n = 21)

6.7.1.2 Correlation Analysis

Table 6.22 presents the correlation coefficients (at the 0.01 significance level) between the structural measures and efficiency—measured by TBT. The results show that the correlation coefficients of the replications are very similar to the original experiment. Therefore, we conclude that, in principle, the results of the correlation analysis seem to be independent of the DIS platform.

6.7.1.3 Simple Regression Analysis

Table 6.23 presents the computed regression coefficients ($\beta$), the actual statistical significance of $\beta$, and indicators ($R$, $R^2$ and adjusted $R^2$) of goodness-of-fit of the

<table>
<thead>
<tr>
<th></th>
<th>TBT₄</th>
<th>TBT₅</th>
<th>TBT₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLOP</td>
<td>0.93</td>
<td>0.94</td>
<td>0.93</td>
</tr>
<tr>
<td>CLIP</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>COOP</td>
<td>0.79</td>
<td>0.79</td>
<td>0.79</td>
</tr>
<tr>
<td>COIP</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>DOP</td>
<td>0.93</td>
<td>0.94</td>
<td>0.93</td>
</tr>
<tr>
<td>DIP</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

Table 6.22: Structural measures and TBT: Spearman's Correlation coefficients (Replications)
Table 6.23: Structural Measures and TBT: Simple Regression Statistics (Replications)

<table>
<thead>
<tr>
<th></th>
<th>Y</th>
<th>X</th>
<th>β₀</th>
<th>β₁</th>
<th>Sig.</th>
<th>R</th>
<th>R²</th>
<th>Adj. R²</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBT₄</td>
<td>CLOP</td>
<td>-0.82</td>
<td>3.49</td>
<td>0.00</td>
<td>0.87</td>
<td>0.76</td>
<td>0.75</td>
<td></td>
</tr>
<tr>
<td>TBT₅</td>
<td>CLOP</td>
<td>-1.10</td>
<td>4.71</td>
<td>0.00</td>
<td>0.86</td>
<td>0.74</td>
<td>0.72</td>
<td></td>
</tr>
<tr>
<td>TBT₂</td>
<td>CLOP</td>
<td>-0.73</td>
<td>3.33</td>
<td>0.00</td>
<td>0.88</td>
<td>0.77</td>
<td>0.76</td>
<td></td>
</tr>
<tr>
<td>TBT₄</td>
<td>COOP</td>
<td>0.00</td>
<td>5.84</td>
<td>0.00</td>
<td>0.63</td>
<td>0.40</td>
<td>0.36</td>
<td></td>
</tr>
<tr>
<td>TBT₅</td>
<td>COOP</td>
<td>0.00</td>
<td>7.89</td>
<td>0.00</td>
<td>0.62</td>
<td>0.38</td>
<td>0.35</td>
<td></td>
</tr>
<tr>
<td>TBT₂</td>
<td>COOP</td>
<td>0.00</td>
<td>5.63</td>
<td>0.00</td>
<td>0.64</td>
<td>0.41</td>
<td>0.38</td>
<td></td>
</tr>
<tr>
<td>TBT₄</td>
<td>DOP</td>
<td>-0.82</td>
<td>3.49</td>
<td>0.00</td>
<td>0.87</td>
<td>0.76</td>
<td>0.75</td>
<td></td>
</tr>
<tr>
<td>TBT₅</td>
<td>DOP</td>
<td>-1.10</td>
<td>4.71</td>
<td>0.00</td>
<td>0.86</td>
<td>0.74</td>
<td>0.72</td>
<td></td>
</tr>
<tr>
<td>TBT₂</td>
<td>DOP</td>
<td>-0.73</td>
<td>3.33</td>
<td>0.00</td>
<td>0.88</td>
<td>0.77</td>
<td>0.76</td>
<td></td>
</tr>
</tbody>
</table>

models. Each row contains the parameter estimates of a simple regression model.

It can be seen that the model goodness-of-fit of the replications are very similar to the original experiment. Consequently, we argue that the results of the simple regression analysis are likely to be independent of the DIS platform.

6.7.1.4 Multiple Regression Analysis

Table 6.24 provides the regression coefficients, their statistical significance, and indicators of goodness-of-fit of the models, after performing stepwise multiple linear regression.

The results show that the model goodness-of-fit of the replications are very similar to the original experiment. Hence we conclude that, in principle, the results of the multiple regression analysis seem to be independent of the DIS platform.

6.7.2 Second Experiment Replications

6.7.2.1 Descriptive statistics

Table 6.25 presents the descriptive statistics of TNF (in bold). A comparison with the original experiment shows that the values are similar to both replications. Furthermore, the distributions of the data are similar to the original experiment.

The histogram plots of the measures suggested that their distributions were not
normal either. This was confirmed by the Shapiro-Wilk test (see Table 6.26). It can be seen that their distribution also deviated significantly from normality.

### 6.7.2.2 Correlation Analysis

Table 6.27 presents the correlation coefficients (at the 0.01 significance level) between the structural measures and reliability—measured by TNF. The results show that the correlations of the replications are very similar to the original experiment. Therefore, we conclude that, in principle, the results of the correlation analysis
Table 6.27: Structural measures and TNF: Spearman’s correlation coefficients (Replications)

<table>
<thead>
<tr>
<th></th>
<th>TNF₁</th>
<th>TNF₂</th>
<th>TNF₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLOP</td>
<td>0.90</td>
<td>0.89</td>
<td>0.89</td>
</tr>
<tr>
<td>CLIP</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>COOP</td>
<td>0.65</td>
<td>0.65</td>
<td>0.65</td>
</tr>
<tr>
<td>COIP</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>DOP</td>
<td>0.90</td>
<td>0.89</td>
<td>0.89</td>
</tr>
<tr>
<td>DIP</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

Table 6.28: Structural Measures and TNF: Simple Regression Statistics (Replications)

<table>
<thead>
<tr>
<th></th>
<th>X</th>
<th>β₀</th>
<th>β₁</th>
<th>Sig.</th>
<th>R</th>
<th>R²</th>
<th>Adj. R²</th>
</tr>
</thead>
<tbody>
<tr>
<td>TNF₁</td>
<td>CLOP</td>
<td>-0.96</td>
<td>3.85</td>
<td>0.00</td>
<td>0.86</td>
<td>0.75</td>
<td>0.73</td>
</tr>
<tr>
<td>TNF₂</td>
<td>CLOP</td>
<td>-0.98</td>
<td>3.84</td>
<td>0.00</td>
<td>0.87</td>
<td>0.75</td>
<td>0.74</td>
</tr>
<tr>
<td>TNF₃</td>
<td>COOP</td>
<td>0.00</td>
<td>6.36</td>
<td>0.00</td>
<td>0.61</td>
<td>0.38</td>
<td>0.34</td>
</tr>
<tr>
<td>TNF₄</td>
<td>COOP</td>
<td>0.00</td>
<td>6.29</td>
<td>0.00</td>
<td>0.61</td>
<td>0.37</td>
<td>0.34</td>
</tr>
<tr>
<td>TNF₅</td>
<td>COOP</td>
<td>0.00</td>
<td>5.92</td>
<td>0.00</td>
<td>0.60</td>
<td>0.36</td>
<td>0.33</td>
</tr>
<tr>
<td>TNF₁</td>
<td>DOP</td>
<td>-0.96</td>
<td>3.85</td>
<td>0.00</td>
<td>0.86</td>
<td>0.75</td>
<td>0.73</td>
</tr>
<tr>
<td>TNF₂</td>
<td>DOP</td>
<td>-0.98</td>
<td>3.84</td>
<td>0.00</td>
<td>0.87</td>
<td>0.75</td>
<td>0.74</td>
</tr>
</tbody>
</table>

It seems to be independent of the DIS platform.

### 6.7.2.3 Simple Regression Analysis

Table 6.28 presents the computed regression coefficients (β), the actual statistical significance of β, and indicators (R, R² and adjusted R²) of goodness-of-fit of the models. Each row contains the parameter estimates of a simple regression model.

It can be seen that the model goodness-of-fit of the replications are very similar to the original experiment. Consequently, we argue that the results of the simple regression analysis are likely to be independent of the DIS platform.
<table>
<thead>
<tr>
<th>( X_i )</th>
<th>( \beta_i )</th>
<th>Sig.</th>
<th>( Y )</th>
<th>( R )</th>
<th>( R^2 )</th>
<th>Adj. ( R^2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>constant</td>
<td>0.00</td>
<td>&gt; 0.05</td>
<td>TNF(_4)</td>
<td>0.90</td>
<td>0.81</td>
<td>0.79</td>
</tr>
<tr>
<td>DOP</td>
<td>5.78</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COOP</td>
<td>-5.20</td>
<td>0.02</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>constant</td>
<td>0.00</td>
<td>&gt; 0.05</td>
<td>TNF(_5)</td>
<td>0.90</td>
<td>0.82</td>
<td>0.80</td>
</tr>
<tr>
<td>DOP</td>
<td>5.81</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COOP</td>
<td>-5.32</td>
<td>0.02</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>constant</td>
<td>0.00</td>
<td>&gt; 0.05</td>
<td>TNF(_2)</td>
<td>0.89</td>
<td>0.80</td>
<td>0.77</td>
</tr>
<tr>
<td>DOP</td>
<td>5.44</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COOP</td>
<td>-4.95</td>
<td>0.03</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.29: Structural Measures and TNF: Multiple Regression Statistics (Replications)

6.7.2.4 Multiple Regression Analysis

Table 6.29 provides the regression coefficients, their statistical significance, and indicators of goodness-of-fit of the models, after performing stepwise multiple linear regression.

The results show that the model goodness-of-fit of the replications are very similar to the original experiment. Hence we conclude that, in principle, the results of the multiple regression analysis seem to be independent of the DIS platform.

6.8 Discussion

The first objective of the experiments was to test whether the structural measures were correlated to the quality attributes of interest. The overall conclusion is that a significant number of the measures are highly correlated to quality attributes. However, it should be noted that some of the measures could not be tested due to the unsuitability of the values obtained for those measures—most values equal to zero. Table 6.30 summarises the relationships among structural measures and quality attributes where:

- 'Y' indicates the relationship was supported by our empirical data,
- 'N' implies the relationship was not supported by our empirical data, and
- '?' indicates the relationship could not be tested with our empirical data.
<table>
<thead>
<tr>
<th></th>
<th>Efficiency</th>
<th>Reliability</th>
<th>Maintainability</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLOP</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>CLOI</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>CLIP</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>CLI2</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>COOP</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>COOI</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>COIP</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>COII</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>DOP</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>DOI</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>DIP</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>DII</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

Table 6.30: Summary of relationships among structural measures and quality attributes

The second objective was to test whether the structural measures can be useful individual estimators of quality attributes. The overall conclusion is that all measures correlated to quality attributes are good individual predictors, except in the case of maintainability. This may be explained by the fact that the relationships were not linear.

The third objective was to test whether individual predictors can be used in combination to estimate quality attributes. We found a statistically significant multiple model with high goodness-of-fit for efficiency and reliability—a multiple model for maintainability was not feasible since only one predictor was available.

These results cannot be considered final. More experiments and replication are necessary to arrive to conclusive results. As a first step in this direction, we replicated the first and the second experiment on different platforms. The changes in the platform (a different operating system and a different network) did not affect the relationships among the structural attributes and the quality attributes. Furthermore, in all cases, the results of the replications were highly consistent with the original experiments. This may be taken as a strong indication that the results are independent of the platform.
6.9 Summary

In this chapter, we have empirically tested the proposed structural measures as indicators of the quality attributes of interest with data extracted from a DIS. The results suggest that a significant number of these measures are correlated to and can be useful estimators of quality attributes. Next chapter presents the empirical evaluation of the behavioural measures.
Chapter 7

Empirical Evaluation – Behavioural Measures

7.1 Introduction

The previous chapter described the first part of the empirical evaluation—i.e. structural measures. The second part of the empirical evaluation (i.e. behavioural measures) is presented in this chapter. The experiments and replications are reported following the guidelines outlined by Wohlin et al. [2000].

7.2 Definition

Following the GQM template [Basili and Rombach, 1988], the experimental goals were defined as shown in Table 7.1.

7.3 Planning

7.3.1 Context

The system studied is an enterprise DIS, entirely written in Java, that supports the academic management of an Australian university. The proof-of-concept system, developed by a team of Computer Science graduates, is composed of 21 distributed components, 67 classes and about 11,000 lines of code. JDBC [Sun, 2003a],
1 Analyse behavioural coupling measures,  
for the purpose of evaluation,  
with respect to their capability of being used as indicators of efficiency,  
from the point of view of DIS software engineers,  
in the context of a proof-of-concept DIS developed by a team of graduates.

2 Analyse behavioural coupling measures,  
for the purpose of evaluation,  
with respect to their capability of being used as indicators of reliability,  
from the point of view of DIS software engineers,  
in the context of a proof-of-concept DIS developed by a team of graduates.

3 Analyse behavioural coupling measures,  
for the purpose of evaluation,  
with respect to their capability of being used as indicators of maintainability,  
from the point of view of DIS software engineers,  
in the context of a proof-of-concept DIS developed by a team of graduates.

Table 7.1: Experimental Goals

JRMI [Sun, 2003c] and JMS [Sun, 2003b] are the middleware technologies used by components to interact.

7.3.2 Experiment Design

It is possible to adopt two main strategies in order to evaluate software measurement hypotheses empirically [Briand et al., 1999]:

- small-scale controlled experiments, and/or
- real-scale industrial case studies.

In this case, for an exploratory investigation, we chose the first alternative since it is more suitable to study the phenomena of interest in isolation, without having to deal with other sources of variation, such as co-existing systems and security mechanisms.

However, we envisage that after several experiments the suite of measures will be shown to be robust, and we intend to test the measures utilising the second strategy—it should be noted that this is outside the scope of this thesis.
7.3.3 Hypotheses

The experimental hypotheses 4, 5 and 6 established in section 4.3.2 are redefined here by refining the design attribute:

- **Hypothesis 4.1.** The behavioural OS coupling of a distributed component affects its extrinsic efficiency.
- **Hypothesis 4.2.** The behavioural OA coupling of a distributed component affects its extrinsic efficiency.
- **Hypothesis 4.3.** The behavioural OC coupling of a distributed component affects its extrinsic efficiency.
- **Hypothesis 4.4.** The behavioural OB coupling of a distributed component affects its extrinsic efficiency.
- **Hypothesis 4.5.** The behavioural IS coupling of a distributed component affects its extrinsic efficiency.
- **Hypothesis 4.6.** The behavioural IA coupling of a distributed component affects its extrinsic efficiency.
- **Hypothesis 4.7.** The behavioural IC coupling of a distributed component affects its extrinsic efficiency.
- **Hypothesis 4.8.** The behavioural IB coupling of a distributed component affects its extrinsic efficiency.
- **Hypothesis 5.1.** The extrinsic reliability of a distributed component is affected by its behavioural OS coupling.
- **Hypothesis 5.2.** The extrinsic reliability of a distributed component is affected by its behavioural OA coupling.
- **Hypothesis 5.3.** The extrinsic reliability of a distributed component is affected by its behavioural OC coupling.
- **Hypothesis 5.4.** The extrinsic reliability of a distributed component is affected by its behavioural OB coupling.
Hypothesis 5.5. The extrinsic reliability of a distributed component is affected by its behavioural IS coupling.

Hypothesis 5.6. The extrinsic reliability of a distributed component is affected by its behavioural IA coupling.

Hypothesis 5.7. The extrinsic reliability of a distributed component is affected by its behavioural IC coupling.

Hypothesis 5.8. The extrinsic reliability of a distributed component is affected by its behavioural IB coupling.

Hypothesis 6.1. The behavioural OS coupling of a distributed component affects its extrinsic maintainability.

Hypothesis 6.2. The behavioural OA coupling of a distributed component affects its extrinsic maintainability.

Hypothesis 6.3. The behavioural OC coupling of a distributed component affects its extrinsic maintainability.

Hypothesis 6.4. The behavioural OB coupling of a distributed component affects its extrinsic maintainability.

Hypothesis 6.5. The behavioural IS coupling of a distributed component affects its extrinsic maintainability.

Hypothesis 6.6. The behavioural IA coupling of a distributed component affects its extrinsic maintainability.

Hypothesis 6.7. The behavioural IC coupling of a distributed component affects its extrinsic maintainability.

Hypothesis 6.8. The behavioural IB coupling of a distributed component affects its extrinsic maintainability.
7.3.4 Variables

Behavioural coupling is quantified by the independent measures defined in Chapter 5 and shown in Table 7.2.

As discussed previously, efficiency, reliability and maintainability are high-level quality attributes that should be evaluated by targeting one of its sub-attributes. Table 7.3 shows the targeted quality sub-attributes, their definitions (according to ISO 9126-1) and the specific dependent measures selected to quantify them. The dependent measures can be described as follows:

- The times \( t_b \) and \( t_a \) are measured immediately before and after each remote interaction, and the difference \( (t_a - t_b) \) is considered the blocking time of the interaction. \( TBT \) is the sum of all blocking times due to remote interactions.

- The occurrence of a failure is recorded whenever a remote interaction throws an exception. \( TNF \) is the sum of all failures due to remote interactions.

- \( OFC \) is calculated as the ratio of \( LET/TET \), where Local Execution Time (\( LET \)) is the difference between Total Execution Time (\( TET \)) and Remote Execution Time (\( RET \)), that is \( LET = TET - RET \). \( RET \) is the time spent satisfying remote interactions. The higher the \( RET \), the lower the \( OFC \), since a component cannot be disconnected from the rest of the system for maintenance while it is interacting with remote components.

7.3.5 Instrumentation

Generic high-level design information from the behaviour of the system was extracted by an electronic tool that we developed—see chapter 8. This information
Table 7.3: Quality Measures

was subsequently used to compute the measures presented above—and eventually others that might be defined.

Measurement data of quality attributes was obtained by the addition of monitoring code to the system in appropriate locations.

7.4 Operation

7.4.1 Preparation

Special code, in addition to the monitoring code, was introduced to some components to simulate the random operation of the system by end-users. Several pilot experiments were run, before the actual experiments, to make sure that there were no apparent anomalies, and that the system behaved in the same way after the monitoring and simulation code was introduced.

In the second experiment, extra code was also added to simulate the occurrence of random faults in the system.

7.4.2 Execution

The system was executed on an isolated network of workstations, each of which equipped with a single CPU and running under the Linux operating system. All
computers had the same hardware and software, and were configured in the same way—the same binary image was used for all hard disks.

Every component was run on a separate workstation as the only user process, all other processes running were a few system processes started by default. The execution of the system was initiated and terminated by the author, who also controlled that in the meantime nobody else had access to the facilities. The experiments were conducted in the Distributed Computing Research Laboratory of RMIT University.

7.4.3 Data validation

The data was thoroughly inspected to assert that it was consistent despite it was collected reliably and objectively by electronic means. For this purpose we ran each experiment at five different times and compared the observed data. There was no need to discard any data, thus all data collected was used.

However, since the results were very similar for the each experiment, only three data sets per experiment are reported to avoid repetition. All combinations produced the same results, therefore we simply omitted the data sets of the last two runs in all cases.

7.5 Analysis and Interpretation of the Results

After the execution of the experiments, all the measures were computed from the recorded data. The empirical data was analysed with the assistance of the software package SPSS [1998]. Raw experimental data can be found in Appendix B.

7.5.1 Descriptive statistics

Table 7.4 and 7.5 present the minimum, maximum, sum, mean, standard deviation and variance of the behavioural measures and the quality measures. The histogram plots of all measures suggested that their distributions were not normal. This was confirmed by the Shapiro-Wilk test. The results show clearly that the distribution of the measures deviated significantly from normality—see Table 7.6 and 7.7.
<table>
<thead>
<tr>
<th>Measure</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Sum</th>
<th>Mean</th>
<th>Std. Dev.</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>COS</td>
<td>0</td>
<td>3</td>
<td>27</td>
<td>1.29</td>
<td>1.19</td>
<td>1.41</td>
</tr>
<tr>
<td>COA</td>
<td>0</td>
<td>3</td>
<td>27</td>
<td>1.29</td>
<td>1.19</td>
<td>1.41</td>
</tr>
<tr>
<td>COC</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>COB</td>
<td>0</td>
<td>2</td>
<td>8</td>
<td>0.38</td>
<td>0.67</td>
<td>0.45</td>
</tr>
<tr>
<td>CIS</td>
<td>0</td>
<td>5</td>
<td>30</td>
<td>1.43</td>
<td>1.63</td>
<td>2.657</td>
</tr>
<tr>
<td>CIA</td>
<td>0</td>
<td>5</td>
<td>30</td>
<td>1.43</td>
<td>1.63</td>
<td>2.66</td>
</tr>
<tr>
<td>CIC</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>CIB</td>
<td>0</td>
<td>2</td>
<td>8</td>
<td>0.38</td>
<td>0.67</td>
<td>0.45</td>
</tr>
<tr>
<td>IOS</td>
<td>0</td>
<td>28</td>
<td>132</td>
<td>6.29</td>
<td>7.71</td>
<td>59.41</td>
</tr>
<tr>
<td>IOA</td>
<td>0</td>
<td>28</td>
<td>132</td>
<td>6.29</td>
<td>7.71</td>
<td>59.41</td>
</tr>
<tr>
<td>IOC</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>IOB</td>
<td>0</td>
<td>5</td>
<td>13</td>
<td>0.62</td>
<td>1.36</td>
<td>1.85</td>
</tr>
<tr>
<td>IIS</td>
<td>0</td>
<td>26</td>
<td>132</td>
<td>6.29</td>
<td>6.97</td>
<td>48.61</td>
</tr>
<tr>
<td>IIA</td>
<td>0</td>
<td>26</td>
<td>132</td>
<td>6.29</td>
<td>6.97</td>
<td>48.61</td>
</tr>
<tr>
<td>IIC</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>IIB</td>
<td>0</td>
<td>4</td>
<td>13</td>
<td>0.62</td>
<td>1.20</td>
<td>1.45</td>
</tr>
<tr>
<td>SOS</td>
<td>0</td>
<td>15</td>
<td>101</td>
<td>4.81</td>
<td>5.23</td>
<td>27.36</td>
</tr>
<tr>
<td>SOA</td>
<td>0</td>
<td>15</td>
<td>101</td>
<td>4.81</td>
<td>5.23</td>
<td>27.36</td>
</tr>
<tr>
<td>SOC</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>SOB</td>
<td>0</td>
<td>15</td>
<td>48</td>
<td>2.29</td>
<td>4.81</td>
<td>23.11</td>
</tr>
<tr>
<td>SIS</td>
<td>0</td>
<td>15</td>
<td>110</td>
<td>5.24</td>
<td>5.55</td>
<td>30.79</td>
</tr>
<tr>
<td>SIA</td>
<td>0</td>
<td>15</td>
<td>110</td>
<td>5.24</td>
<td>5.55</td>
<td>30.79</td>
</tr>
<tr>
<td>SIC</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>SIB</td>
<td>0</td>
<td>15</td>
<td>51</td>
<td>2.43</td>
<td>4.85</td>
<td>23.56</td>
</tr>
<tr>
<td>SL</td>
<td>0</td>
<td>3</td>
<td>30</td>
<td>1.43</td>
<td>1.21</td>
<td>1.46</td>
</tr>
<tr>
<td>AL</td>
<td>0</td>
<td>3</td>
<td>30</td>
<td>1.43</td>
<td>1.21</td>
<td>1.46</td>
</tr>
<tr>
<td>CL</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>BL</td>
<td>0</td>
<td>1</td>
<td>6</td>
<td>0.29</td>
<td>0.46</td>
<td>0.21</td>
</tr>
</tbody>
</table>

Table 7.4: Behavioural Measures: Descriptive Statistics (n = 21)
### 7.5. Analysis and Interpretation of the Results

<table>
<thead>
<tr>
<th>Measure</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Sum</th>
<th>Mean</th>
<th>Std. Dev.</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBT₁</td>
<td>0.00</td>
<td>11.57</td>
<td>85.90</td>
<td>4.09</td>
<td>4.14</td>
<td>17.14</td>
</tr>
<tr>
<td>TBT₂</td>
<td>0.00</td>
<td>11.59</td>
<td>84.51</td>
<td>4.02</td>
<td>4.07</td>
<td>16.60</td>
</tr>
<tr>
<td>TBT₃</td>
<td>0.00</td>
<td>11.83</td>
<td>86.46</td>
<td>4.12</td>
<td>4.18</td>
<td>17.45</td>
</tr>
<tr>
<td>TNF₁</td>
<td>0.00</td>
<td>12.44</td>
<td>88.67</td>
<td>4.22</td>
<td>4.61</td>
<td>21.22</td>
</tr>
<tr>
<td>TNF₂</td>
<td>0.00</td>
<td>12.11</td>
<td>88.84</td>
<td>4.23</td>
<td>4.54</td>
<td>20.57</td>
</tr>
<tr>
<td>TNF₃</td>
<td>0.00</td>
<td>12.31</td>
<td>92.30</td>
<td>4.39</td>
<td>4.67</td>
<td>21.79</td>
</tr>
<tr>
<td>OFC₁</td>
<td>31.93</td>
<td>100.00</td>
<td>1720.40</td>
<td>81.92</td>
<td>23.56</td>
<td>554.91</td>
</tr>
<tr>
<td>OFC₂</td>
<td>35.27</td>
<td>100.00</td>
<td>1725.87</td>
<td>82.18</td>
<td>23.03</td>
<td>530.44</td>
</tr>
<tr>
<td>OFC₃</td>
<td>35.27</td>
<td>100.00</td>
<td>1725.87</td>
<td>82.18</td>
<td>23.03</td>
<td>530.44</td>
</tr>
</tbody>
</table>

Table 7.5: Quality Measures: Descriptive Statistics ($n = 21$)

<table>
<thead>
<tr>
<th>Measure</th>
<th>Shapiro-Wilk Statistic</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>COS</td>
<td>0.765</td>
<td>0.000</td>
</tr>
<tr>
<td>COA</td>
<td>0.765</td>
<td>0.000</td>
</tr>
<tr>
<td>COB</td>
<td>0.617</td>
<td>0.000</td>
</tr>
<tr>
<td>CIS</td>
<td>0.736</td>
<td>0.000</td>
</tr>
<tr>
<td>CIA</td>
<td>0.736</td>
<td>0.000</td>
</tr>
<tr>
<td>CIB</td>
<td>0.617</td>
<td>0.000</td>
</tr>
<tr>
<td>IOS</td>
<td>0.796</td>
<td>0.001</td>
</tr>
<tr>
<td>IOA</td>
<td>0.796</td>
<td>0.001</td>
</tr>
<tr>
<td>IOB</td>
<td>0.521</td>
<td>0.000</td>
</tr>
<tr>
<td>IIS</td>
<td>0.840</td>
<td>0.003</td>
</tr>
<tr>
<td>IIA</td>
<td>0.840</td>
<td>0.003</td>
</tr>
<tr>
<td>IIB</td>
<td>0.584</td>
<td>0.000</td>
</tr>
<tr>
<td>SOS</td>
<td>0.836</td>
<td>0.002</td>
</tr>
<tr>
<td>SOA</td>
<td>0.836</td>
<td>0.002</td>
</tr>
<tr>
<td>SOB</td>
<td>0.538</td>
<td>0.000</td>
</tr>
<tr>
<td>SIS</td>
<td>0.826</td>
<td>0.002</td>
</tr>
<tr>
<td>SIA</td>
<td>0.826</td>
<td>0.002</td>
</tr>
<tr>
<td>SIB</td>
<td>0.569</td>
<td>0.000</td>
</tr>
<tr>
<td>SL</td>
<td>0.835</td>
<td>0.002</td>
</tr>
<tr>
<td>AL</td>
<td>0.835</td>
<td>0.002</td>
</tr>
<tr>
<td>BL</td>
<td>0.570</td>
<td>0.000</td>
</tr>
</tbody>
</table>

Table 7.6: Behavioural Measures: Normality Test ($n = 21$)
Most behavioural measures are simple counts. Hence, their values are non-negative integer numbers and their minimum value is zero. All behavioural measures were defined and theoretically validated on the ratio scale in Chapter 5. The observations corresponding to the measures COC, CIC, IOC, IIC, SOC, SIC and CL had almost all their values equal to zero. Therefore they were excluded from the correlation analysis—a similar approach is followed by El Emam et al. [2001].

TBT is a measure of time and its units are hundreds of seconds. TNF is a non-negative integer and its units are hundreds of failures. OFC is a percentage, hence a non-negative real number less than or equal to 100.

It is important to remark that the different measurements of TBT, TNF and OFC are highly consistent. That is, it can be seen that similar data are observed from each run of the experiments.

### 7.5.2 Correlation Analysis

As the data from all the measures was not normally distributed, Spearman, a non-parametric statistic was used. Non-significant correlation coefficients are not shown, they have been substituted by asterisks (*). These measures will not be considered in the regression analysis.
7.5.2.1 Efficiency

Table 7.8 presents the correlation coefficients (at the 0.01 significance level) between the behavioural measures and TBT. The results show that the correlations of TBT with the measures of OS and OA coupling (namely COS, IOS, COS, SL, COA, IOA, SOA, AL) are statistically significant. Furthermore, these correlation coefficients are highly significant, indicating a nontrivial relationship of these measures with TBT. This also suggests that these variables are candidates for a base regression model to estimate efficiency in terms of TBT. Examination of the correlation coefficients indicates that these eight measures are (positively) correlated to TBT. It should be noted that a higher value of TBT indicates worse efficiency.

Therefore, the hypotheses (4.1 and 4.2) underlying the measures of OS and OA coupling seem to be supported in our environment. Components with higher OS and OA coupling exhibit higher values of TBT, and hence worse efficiency. On the other hand, the correlations of the measures of OB, IB, IS and IA coupling are not statistically significant. Hence, hypotheses 4.4, 4.5, 4.6 and 4.8 are not supported in our context. Due to unsuitability of the available data of the OC and IC coupling measures, we were unable to test hypotheses 4.3 and 4.7.

These findings cannot be considered final, they can only be considered preliminary. Further experimentation is indispensable to extract conclusive results. It is necessary to replicate the experiment and to carry out new ones to confirm the tested hypotheses and to evaluate the untested hypotheses.

7.5.2.2 Reliability

The correlation coefficients (at the 0.01 significance level) between the behavioural measures and TNF are presented in Table 7.9. The results show that the correlations of the measures of OS and OA coupling (namely COS, IOS, SOS, SL, COA, IOA, SOA, AL) with TNF are statistically significant. Furthermore, these correlation coefficients are highly significant, indicating a nontrivial association of these measures with TNF. This also suggests that these variables are candidates for a base regression model to estimate reliability in terms of TNF. Examination of the correlation coefficients indicates that these eight measures are (positively) correlated to
<table>
<thead>
<tr>
<th></th>
<th>TBT₁</th>
<th>TBT₂</th>
<th>TBT₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>COS</td>
<td>0.90</td>
<td>0.90</td>
<td>0.90</td>
</tr>
<tr>
<td>COA</td>
<td>0.90</td>
<td>0.90</td>
<td>0.90</td>
</tr>
<tr>
<td>COB</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>CIS</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>CIA</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>CIB</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>IOS</td>
<td>0.95</td>
<td>0.94</td>
<td>0.95</td>
</tr>
<tr>
<td>IOA</td>
<td>0.95</td>
<td>0.94</td>
<td>0.95</td>
</tr>
<tr>
<td>IOB</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>IIS</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>IIA</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>IIB</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>SOS</td>
<td>0.84</td>
<td>0.83</td>
<td>0.84</td>
</tr>
<tr>
<td>SOA</td>
<td>0.84</td>
<td>0.83</td>
<td>0.84</td>
</tr>
<tr>
<td>SOB</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>SIS</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>SIA</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>SIB</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>SL</td>
<td>0.81</td>
<td>0.82</td>
<td>0.81</td>
</tr>
<tr>
<td>AL</td>
<td>0.81</td>
<td>0.82</td>
<td>0.81</td>
</tr>
<tr>
<td>BL</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

Table 7.8: Behavioural measures and TBT: Correlation coefficients
TNF. It should be noted that a higher value of TNF indicates worse reliability.

Consequently, the hypotheses (5.1 and 5.2) underlying the measures of OS and OA coupling seem to be supported in our environment. Components with higher OS and OA coupling exhibit higher values of TNF, and hence worse reliability. On the other hand, the correlations of the measures of OB, IB, IS, AI coupling are not statistically significant. Therefore, hypotheses 5.4, 5.5, 5.6 and 5.8 are not supported in our environment. Due to unsuitability of the available data of the OC and IC coupling measures, we were unable to test hypotheses 5.3 and 5.7.

These findings cannot be considered final, they can only be considered preliminary. Further experimentation is indispensable to extract conclusive results. It is necessary to replicate the experiment and to carry out new ones to confirm the tested hypotheses and to evaluate the untested hypotheses.

7.5.2.3 Maintainability

Table 7.10 presents the correlation coefficients (at the 0.01 significance level) between the behavioural measures and OFC. The results show that the correlations of the measures of IS and IA coupling (namely CIS, CIA, IIS, IIA, SIS and SIA) with OFC are statistically significant. Furthermore, these correlation coefficients are highly significant, indicating a nontrivial association of these measures with OFC. This also suggests that these variables are candidates for a base regression model to estimate maintainability, in terms of OFC. Examination of the correlation coefficients indicates that these six measures are (negatively) correlated to OFC.

Therefore, the hypotheses (6.5 and 6.6) underlying the measures of IS and IA coupling seem to be supported in our environment. Components with higher IS and IA coupling exhibit lower values of OFC, and hence worse maintainability. On the other hand, the correlations of the measures of IB, OB, OS and OA coupling are not statistically significant. Hence, hypotheses 6.1, 6.2, 6.4 and 6.8 are not supported in our environment. Due to unsuitability of the available data of the OC and IC coupling measures, we were unable to test hypotheses 6.3 and 6.7.

These findings cannot be considered final, they can only be considered preliminary. Further experimentation is indispensable to extract conclusive results. It
<table>
<thead>
<tr>
<th></th>
<th>TNF₁</th>
<th>TNF₂</th>
<th>TNF₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>COS</td>
<td>0.911</td>
<td>0.920</td>
<td>0.920</td>
</tr>
<tr>
<td>COA</td>
<td>0.911</td>
<td>0.920</td>
<td>0.920</td>
</tr>
<tr>
<td>COB</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>CIS</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>CIA</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>CIB</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>IOS</td>
<td>0.917</td>
<td>0.923</td>
<td>0.917</td>
</tr>
<tr>
<td>IOA</td>
<td>0.917</td>
<td>0.923</td>
<td>0.917</td>
</tr>
<tr>
<td>IOB</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>IIS</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>IIA</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>IIB</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>SOS</td>
<td>0.870</td>
<td>0.872</td>
<td>0.861</td>
</tr>
<tr>
<td>SOA</td>
<td>0.870</td>
<td>0.872</td>
<td>0.861</td>
</tr>
<tr>
<td>SOB</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>SIS</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>SIA</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>SIB</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>SL</td>
<td>0.796</td>
<td>0.777</td>
<td>0.789</td>
</tr>
<tr>
<td>AL</td>
<td>0.796</td>
<td>0.777</td>
<td>0.789</td>
</tr>
<tr>
<td>BL</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

Table 7.9: Behavioural measures and TNF: Correlation coefficients
is necessary to replicate the experiment and to carry out new ones to confirm the tested hypotheses and to evaluate the untested hypotheses.

### 7.5.3 Simple Regression Analysis

In this section, we present the results obtained when analysing the individual impact of the behavioural measures on quality attributes using regression analysis. As discussed in Chapter 3, a multiple linear regression equation has the following form:

$$ Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n + \varepsilon $$  \hspace{1cm} (7.1)
where \( Y \) is the response variable, \( X_i \) are the explanatory variables, and \( \beta_i \) are the regression coefficients. A simple regression model is a special case of this, where only one explanatory variable appears.

Table 7.11 presents the computed regression coefficients (\( \beta \)), the actual statistical significance of \( \beta \), and indicators (\( R, R^2 \) and adjusted \( R^2 \)) of goodness-of-fit of the efficiency models. Each row contains the parameter estimates of a simple regression model. The results obtained indicate that the measures considered in this section (namely COS, IOS, SOS, SL, COA, IOA, SOA, AL) not only correlate with efficiency, but are also useful to explain the variation in TBT. For example, in our environment, OS coupling (measured by COS) accounted for 82 percent of the variation in efficiency (measured by TBT), and each increase of one unit of COS increased the TBT by 301 seconds.

The computed regression coefficients (\( \beta \)), the actual statistical significance of \( \beta \), and indicators (\( R, R^2 \) and adjusted \( R^2 \)) of goodness-of-fit of the reliability models are presented in Table 7.12. Each row contains the parameter estimates of a simple regression model. The results obtained indicate that the measures considered in this section (namely COS, IOS, SOS, SL, COA, IOA, SOA, AL) not only correlate with reliability, but are also useful to explain the variation in TNF. For example, in our environment, OA coupling (measured by COA) accounted for 79 percent of the variation in reliability (measured by TNF), and each increase of one unit of COA increased the TNF by 340 failures.

Table 7.13 presents the computed regression coefficients (\( \beta \)), the actual statistical significance of \( \beta \), and indicators (\( R, R^2 \) and adjusted \( R^2 \)) of goodness-of-fit of the maintainability models. Each row contains the parameter estimates of a simple regression model. The results obtained indicate that most of the measures considered in this section (namely IIS, IIA, SIS and SIA) not only correlate with maintainability, but are also useful to explain the variation in OFC. For example, in our environment, IS coupling (measured by SIS) accounted for 68 percent of the variation in maintainability (measured by OFC), and each increase of one unit of SIS decreased the OFC by 3.5 units.
<table>
<thead>
<tr>
<th>$Y$</th>
<th>$X$</th>
<th>$\beta_0$</th>
<th>$\beta_1$</th>
<th>Sig.</th>
<th>$R$</th>
<th>$R^2$</th>
<th>Adj. $R^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBT$_1$</td>
<td>COS</td>
<td>0.05</td>
<td>3.15</td>
<td>0.00</td>
<td>0.90</td>
<td>0.82</td>
<td>0.81</td>
</tr>
<tr>
<td>TBT$_2$</td>
<td>COS</td>
<td>0.04</td>
<td>3.01</td>
<td>0.00</td>
<td>0.91</td>
<td>0.82</td>
<td>0.81</td>
</tr>
<tr>
<td>TBT$_3$</td>
<td>COS</td>
<td>0.03</td>
<td>3.18</td>
<td>0.00</td>
<td>0.90</td>
<td>0.82</td>
<td>0.81</td>
</tr>
<tr>
<td>TBT$_1$</td>
<td>COA</td>
<td>0.05</td>
<td>3.15</td>
<td>0.00</td>
<td>0.90</td>
<td>0.82</td>
<td>0.81</td>
</tr>
<tr>
<td>TBT$_2$</td>
<td>COA</td>
<td>0.04</td>
<td>3.01</td>
<td>0.00</td>
<td>0.91</td>
<td>0.82</td>
<td>0.81</td>
</tr>
<tr>
<td>TBT$_3$</td>
<td>COA</td>
<td>0.03</td>
<td>3.18</td>
<td>0.00</td>
<td>0.90</td>
<td>0.82</td>
<td>0.81</td>
</tr>
<tr>
<td>TBT$_1$</td>
<td>IOS</td>
<td>1.19</td>
<td>0.46</td>
<td>0.00</td>
<td>0.86</td>
<td>0.74</td>
<td>0.72</td>
</tr>
<tr>
<td>TBT$_2$</td>
<td>IOS</td>
<td>1.14</td>
<td>0.44</td>
<td>0.00</td>
<td>0.86</td>
<td>0.74</td>
<td>0.72</td>
</tr>
<tr>
<td>TBT$_3$</td>
<td>IOS</td>
<td>1.18</td>
<td>0.47</td>
<td>0.00</td>
<td>0.86</td>
<td>0.74</td>
<td>0.73</td>
</tr>
<tr>
<td>TBT$_1$</td>
<td>IOA</td>
<td>1.19</td>
<td>0.46</td>
<td>0.00</td>
<td>0.86</td>
<td>0.74</td>
<td>0.72</td>
</tr>
<tr>
<td>TBT$_2$</td>
<td>IOA</td>
<td>1.14</td>
<td>0.44</td>
<td>0.00</td>
<td>0.86</td>
<td>0.74</td>
<td>0.72</td>
</tr>
<tr>
<td>TBT$_3$</td>
<td>IOA</td>
<td>1.18</td>
<td>0.47</td>
<td>0.00</td>
<td>0.86</td>
<td>0.74</td>
<td>0.73</td>
</tr>
<tr>
<td>TBT$_1$</td>
<td>SOS</td>
<td>1.15</td>
<td>0.61</td>
<td>0.00</td>
<td>0.77</td>
<td>0.60</td>
<td>0.57</td>
</tr>
<tr>
<td>TBT$_2$</td>
<td>SOS</td>
<td>1.13</td>
<td>0.58</td>
<td>0.00</td>
<td>0.76</td>
<td>0.58</td>
<td>0.56</td>
</tr>
<tr>
<td>TBT$_3$</td>
<td>SOS</td>
<td>1.17</td>
<td>0.61</td>
<td>0.00</td>
<td>0.77</td>
<td>0.59</td>
<td>0.57</td>
</tr>
<tr>
<td>TBT$_1$</td>
<td>SOA</td>
<td>1.15</td>
<td>0.61</td>
<td>0.00</td>
<td>0.77</td>
<td>0.60</td>
<td>0.57</td>
</tr>
<tr>
<td>TBT$_2$</td>
<td>SOA</td>
<td>1.13</td>
<td>0.58</td>
<td>0.00</td>
<td>0.76</td>
<td>0.58</td>
<td>0.56</td>
</tr>
<tr>
<td>TBT$_3$</td>
<td>SOA</td>
<td>1.17</td>
<td>0.61</td>
<td>0.00</td>
<td>0.77</td>
<td>0.59</td>
<td>0.57</td>
</tr>
<tr>
<td>TBT$_1$</td>
<td>SL</td>
<td>0.51</td>
<td>2.51</td>
<td>0.00</td>
<td>0.73</td>
<td>0.53</td>
<td>0.51</td>
</tr>
<tr>
<td>TBT$_2$</td>
<td>SL</td>
<td>0.45</td>
<td>2.42</td>
<td>0.00</td>
<td>0.74</td>
<td>0.55</td>
<td>0.52</td>
</tr>
<tr>
<td>TBT$_3$</td>
<td>SL</td>
<td>0.49</td>
<td>2.54</td>
<td>0.00</td>
<td>0.74</td>
<td>0.54</td>
<td>0.51</td>
</tr>
<tr>
<td>TBT$_1$</td>
<td>AL</td>
<td>0.51</td>
<td>2.51</td>
<td>0.00</td>
<td>0.73</td>
<td>0.53</td>
<td>0.51</td>
</tr>
<tr>
<td>TBT$_2$</td>
<td>AL</td>
<td>0.45</td>
<td>2.42</td>
<td>0.00</td>
<td>0.74</td>
<td>0.55</td>
<td>0.52</td>
</tr>
<tr>
<td>TBT$_3$</td>
<td>AL</td>
<td>0.49</td>
<td>2.54</td>
<td>0.00</td>
<td>0.74</td>
<td>0.54</td>
<td>0.51</td>
</tr>
</tbody>
</table>

Table 7.11: Behavioural Measures and TBT: Simple Regression Statistics
### Table 7.12: Behavioural Measures and TNF: Simple Regression Statistics

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$\beta_0$</td>
<td>$\beta_1$</td>
<td>Sig.</td>
<td>$R$</td>
</tr>
<tr>
<td><strong>TNF1</strong></td>
<td>COS</td>
<td>-0.14</td>
<td>3.39</td>
<td>0.00</td>
<td>0.88</td>
</tr>
<tr>
<td><strong>TNF2</strong></td>
<td>COS</td>
<td>-0.14</td>
<td>3.40</td>
<td>0.00</td>
<td>0.89</td>
</tr>
<tr>
<td><strong>TNF3</strong></td>
<td>COS</td>
<td>-0.14</td>
<td>3.53</td>
<td>0.00</td>
<td>0.90</td>
</tr>
<tr>
<td><strong>TNF1</strong></td>
<td>COA</td>
<td>-0.14</td>
<td>3.39</td>
<td>0.00</td>
<td>0.88</td>
</tr>
<tr>
<td><strong>TNF2</strong></td>
<td>COA</td>
<td>-0.14</td>
<td>3.40</td>
<td>0.00</td>
<td>0.89</td>
</tr>
<tr>
<td><strong>TNF3</strong></td>
<td>COA</td>
<td>-0.14</td>
<td>3.53</td>
<td>0.00</td>
<td>0.90</td>
</tr>
<tr>
<td><strong>TNF1</strong></td>
<td>IOS</td>
<td>1.02</td>
<td>0.51</td>
<td>0.00</td>
<td>0.85</td>
</tr>
<tr>
<td><strong>TNF2</strong></td>
<td>IOS</td>
<td>1.09</td>
<td>0.50</td>
<td>0.00</td>
<td>0.85</td>
</tr>
<tr>
<td><strong>TNF3</strong></td>
<td>IOS</td>
<td>1.15</td>
<td>0.52</td>
<td>0.00</td>
<td>0.85</td>
</tr>
<tr>
<td><strong>TNF1</strong></td>
<td>IOA</td>
<td>1.02</td>
<td>0.51</td>
<td>0.00</td>
<td>0.85</td>
</tr>
<tr>
<td><strong>TNF2</strong></td>
<td>IOA</td>
<td>1.09</td>
<td>0.50</td>
<td>0.00</td>
<td>0.85</td>
</tr>
<tr>
<td><strong>TNF3</strong></td>
<td>IOA</td>
<td>1.15</td>
<td>0.52</td>
<td>0.00</td>
<td>0.85</td>
</tr>
<tr>
<td><strong>TNF1</strong></td>
<td>SOS</td>
<td>0.92</td>
<td>0.69</td>
<td>0.00</td>
<td>0.78</td>
</tr>
<tr>
<td><strong>TNF2</strong></td>
<td>SOS</td>
<td>1.02</td>
<td>0.67</td>
<td>0.00</td>
<td>0.77</td>
</tr>
<tr>
<td><strong>TNF3</strong></td>
<td>SOS</td>
<td>1.08</td>
<td>0.69</td>
<td>0.00</td>
<td>0.77</td>
</tr>
<tr>
<td><strong>TNF1</strong></td>
<td>SOA</td>
<td>0.92</td>
<td>0.69</td>
<td>0.00</td>
<td>0.78</td>
</tr>
<tr>
<td><strong>TNF2</strong></td>
<td>SOA</td>
<td>1.02</td>
<td>0.67</td>
<td>0.00</td>
<td>0.77</td>
</tr>
<tr>
<td><strong>TNF3</strong></td>
<td>SOA</td>
<td>1.08</td>
<td>0.69</td>
<td>0.00</td>
<td>0.77</td>
</tr>
<tr>
<td><strong>TNF1</strong></td>
<td>SL</td>
<td>0.28</td>
<td>2.77</td>
<td>0.00</td>
<td>0.72</td>
</tr>
<tr>
<td><strong>TNF2</strong></td>
<td>SL</td>
<td>0.31</td>
<td>2.74</td>
<td>0.00</td>
<td>0.73</td>
</tr>
<tr>
<td><strong>TNF3</strong></td>
<td>SL</td>
<td>0.36</td>
<td>2.83</td>
<td>0.00</td>
<td>0.73</td>
</tr>
<tr>
<td><strong>TNF1</strong></td>
<td>AL</td>
<td>0.28</td>
<td>2.77</td>
<td>0.00</td>
<td>0.72</td>
</tr>
<tr>
<td><strong>TNF2</strong></td>
<td>AL</td>
<td>0.31</td>
<td>2.74</td>
<td>0.00</td>
<td>0.73</td>
</tr>
<tr>
<td><strong>TNF3</strong></td>
<td>AL</td>
<td>0.36</td>
<td>2.83</td>
<td>0.00</td>
<td>0.73</td>
</tr>
</tbody>
</table>

*Note: The table shows the coefficients ($\beta_0$, $\beta_1$), significance (Sig.), correlation coefficient ($R$), coefficient of determination ($R^2$), and adjusted coefficient of determination (Adj. $R^2$) for the simple linear regression models between different measures and the target variable 'TNF'.*
### 7.5.4 Multiple Regression Analysis

It is important to note that we do not expect the behavioural measures to explain all the variation of efficiency, since other factors are likely to be important too, such as network bandwidth and system load. However, the purpose of multiple regression analysis is to determine whether the measures appearing significant in the simple regression analysis are complementary.

Furthermore, in this dissertation we are interested in the goodness-of-fit of the models, and did not investigate the predictive capability of the models per se, although a satisfactory goodness of fit is necessary in order to expect a satisfactory predictive capability in future studies.

The analysis of residuals is a simple yet powerful tool for evaluating the appropriateness of the multiple regression models [Freund and Wilson, 1998]. Consequently, the underlying assumptions will be evaluated:

- **Homogeneity of the error term variance**: The variance of the error terms should not be very unequal.

- **Independence of the error term**: The error terms should not be strongly correlated.
Table 7.14: Behavioural Measures and TBT: Multiple Regression Statistics

- Normality of the error term distribution: The errors should be random and normally distributed.

7.5.4.1 Efficiency

Table 7.14 provides the estimated regression coefficients ($\beta$) and their significance ($\rho$) based on a $t$ test, for two efficiency models found, after performing stepwise multiple linear regression [Freund and Wilson, 1998]. Multiple regression models show a better goodness-of-fit than simple regression models, so the measures (in combination) are deemed to be potentially useful for building a model to predict efficiency. The fact that the significance for $\beta_0$ is greater than 0.05 only means that we cannot conclude that $\beta_0 \neq 0$. However, we can conclude that is very unlikely that $\beta_i = 0$ with $i = 1, 2$. Thus, the statistical evidence shows that it is very likely that IOS, SL, SOA and AL are correlated to efficiency.
7.5. ANALYSIS AND INTERPRETATION OF THE RESULTS

<table>
<thead>
<tr>
<th></th>
<th>Y = TBT₁</th>
<th>Y = TBT₂</th>
<th>Y = TBT₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>X₁ = IOS, X₂ = SL</td>
<td>1.787</td>
<td>1.849</td>
<td>1.844</td>
</tr>
<tr>
<td>X₁ = SOA, X₂ = AL</td>
<td>2.012</td>
<td>2.063</td>
<td>2.121</td>
</tr>
</tbody>
</table>

Table 7.15: Independence of the residuals (TBT): Durbin-Watson Test

<table>
<thead>
<tr>
<th></th>
<th>Y = TBT₁</th>
<th>Y = TBT₂</th>
<th>Y = TBT₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>X₁ = IOS, X₂ = SL</td>
<td>Statistic 0.959</td>
<td>0.954</td>
<td>0.956</td>
</tr>
<tr>
<td></td>
<td>Significance 0.501</td>
<td>0.407</td>
<td>0.448</td>
</tr>
<tr>
<td>X₁ = SOA, X₂ = AL</td>
<td>Statistic 0.857</td>
<td>0.856</td>
<td>0.857</td>
</tr>
<tr>
<td></td>
<td>Significance 0.006</td>
<td>0.005</td>
<td>0.006</td>
</tr>
</tbody>
</table>

Table 7.16: Normality of the residuals (TBT): Shapiro-Wilk Test

The underlying assumptions have been evaluated as follows:

- **Homogeneity of the error term variance.** This assumption holds since the plots of standardized residuals against the standardized predicted values (e.g. Figures 7.1 and 7.2) show a random scatter of points with no discernible pattern identified.

- **Independence of the error term.** We tested this assumption with a Durbin-Watson test that did not reveal a significant correlation between residuals (see Table 7.15). This statistic can vary between 0 and 4, with values between 1.5 and 2.5 indicating that residuals are not correlated strongly.

- **Normality of the error term distribution.** In one case, the results of the Shapiro-Wilk test are not significant (α = 0.05), therefore we can conclude that this assumption is tenable (see Table 7.16). In the other case, there is lack of normality. Nevertheless, the model is robust with respect to this assumption, thus we can accept conclusions based on this model.

Finally, DOP is not included in the models because it is highly correlated to CLOP. Therefore, since the inclusion of DOP to the model does not increase R, this variable in considered redundant.
Figure 7.1: Scatter plot of residuals versus predicted values \((Y = TBT_2, X_1 = I0S, X_2 = SL)\)
Figure 7.2: Scatter plot of residuals versus predicted values ($Y = TBT_2, X_1 = SOA, X_2 = AL$)
### Chapter 7. Empirical Evaluation – Behavioural Measures

<table>
<thead>
<tr>
<th>$X_i$</th>
<th>$\beta_i$</th>
<th>Sig.</th>
<th>Y</th>
<th>$R$</th>
<th>$R^2$</th>
<th>Adj. $R^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>constant</td>
<td>-0.41</td>
<td>&gt; 0.05</td>
<td>TNF$_1$</td>
<td>0.92</td>
<td>0.85</td>
<td>0.83</td>
</tr>
<tr>
<td>IOA</td>
<td>0.39</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AL</td>
<td>1.52</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>constant</td>
<td>-0.36</td>
<td>&gt; 0.05</td>
<td>TNF$_2$</td>
<td>0.92</td>
<td>0.85</td>
<td>0.83</td>
</tr>
<tr>
<td>IOA</td>
<td>0.38</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AL</td>
<td>1.54</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>constant</td>
<td>-0.33</td>
<td>&gt; 0.05</td>
<td>TNF$_3$</td>
<td>0.92</td>
<td>0.85</td>
<td>0.84</td>
</tr>
<tr>
<td>IOA</td>
<td>0.39</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AL</td>
<td>1.58</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>constant</td>
<td>-0.65</td>
<td>&gt; 0.05</td>
<td>TNF$_1$</td>
<td>0.88</td>
<td>0.77</td>
<td>0.74</td>
</tr>
<tr>
<td>SOS</td>
<td>0.49</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL</td>
<td>1.74</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>constant</td>
<td>-0.58</td>
<td>&gt; 0.05</td>
<td>TNF$_2$</td>
<td>0.87</td>
<td>0.77</td>
<td>0.74</td>
</tr>
<tr>
<td>SOS</td>
<td>0.47</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL</td>
<td>1.77</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>constant</td>
<td>-0.57</td>
<td>&gt; 0.05</td>
<td>TNF$_3$</td>
<td>0.88</td>
<td>0.77</td>
<td>0.74</td>
</tr>
<tr>
<td>SOS</td>
<td>0.49</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL</td>
<td>1.82</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7.17: Behavioural Measures and TNF: Multiple Regression Statistics

#### 7.5.4.2 Reliability

Table 7.17 provides the estimated regression coefficients ($\beta$) and their significance ($\rho$) based on a $t$ test, for the reliability models after performing stepwise multiple linear regression [Freund and Wilson, 1998]. Multiple regression models show a better goodness-of-fit than simple regression models, so the measures (in combination) are deemed to be potentially useful for building a model to predict reliability. The fact that the significance for $\beta_0$ is greater than 0.05 only means that we cannot conclude that $\beta_0 \neq 0$. However, we can conclude that is very unlikely that $\beta_i = 0$ with $i = 1, 2$. Thus, the statistical evidence shows that it is very likely that IOA, AL, SOS and SL are correlated to reliability.

The underlying assumptions have been evaluated as follows:

- **Homogeneity of the error term variance.** This assumption holds since the plots
of standardized residuals against the standardized predicted values (e.g. Figures 7.3 and 7.4) show a random scatter of points with no discernible pattern identified.

- **Independence of the error term.** We tested this assumption with a Durbin-Watson test that did not reveal a significant correlation between residuals (see Table 7.18). This statistic can vary between 0 and 4 with values between 1.5 and 2.5 indicating that residuals are not correlated seriously.

- **Normality of the error term distribution.** The results of the Shapiro-Wilk test are not significant ($\alpha = 0.05$), therefore we can conclude that this assumption is tenable (see Table 7.19).

Finally, DOP is not included in the models because it is highly correlated to
Figure 7.4: Scatter plot of residuals versus predicted values ($Y = TNF_2$, $X_1 = SOS$, $X_2 = SL$)

<table>
<thead>
<tr>
<th></th>
<th>$Y = TNF_1$</th>
<th>$Y = TNF_2$</th>
<th>$Y = TNF_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1 = IOA$, $X_2 = AL$</td>
<td>1.584</td>
<td>1.513</td>
<td>1.492</td>
</tr>
<tr>
<td>$X_1 = SOS$, $X_2 = SL$</td>
<td>1.816</td>
<td>1.736</td>
<td>1.738</td>
</tr>
</tbody>
</table>

Table 7.18: Independence of the residuals (TNF): Durbin-Watson Test

<table>
<thead>
<tr>
<th></th>
<th>$Y = TNF_1$</th>
<th>$Y = TNF_2$</th>
<th>$Y = TNF_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1 = IOA$, $X_2 = AL$</td>
<td>Statistic</td>
<td>0.920</td>
<td>0.938</td>
</tr>
<tr>
<td></td>
<td>Significance</td>
<td>0.085</td>
<td>0.195</td>
</tr>
<tr>
<td>$X_1 = SOS$, $X_2 = SL$</td>
<td>Statistic</td>
<td>0.955</td>
<td>0.942</td>
</tr>
<tr>
<td></td>
<td>Significance</td>
<td>0.427</td>
<td>0.241</td>
</tr>
</tbody>
</table>

Table 7.19: Normality of the residuals (TNF): Shapiro-Wilk Test
CLOP. Therefore, since the inclusion of DOP to the model does not increase $R$, this variable in considered redundant.

### 7.5.4.3 Maintainability

Table 7.20 provides the estimated regression coefficients ($\beta$) and their significance ($\rho$) based on a $t$ test, for a maintainability models found, after performing stepwise multiple linear regression [Freund and Wilson, 1998]. Multiple regression models show a better goodness-of-fit than simple regression models, so the measures (in combination) are deemed to be potentially useful for building a model to predict maintainability. The fact that the significance for $\beta_0$ is greater than 0.05 only means that we cannot conclude that $\beta_0 \neq 0$. However, we can conclude that is very unlikely that $\beta_i = 0$ with $i = 1, 2$. Thus, the statistical evidence shows that it is very likely that IOS, SL, SOA and AL are correlated to maintainability.

The underlying assumptions have been evaluated as follows:

- *Homogeneity of the error term variance.* This assumption holds since the plots of standardized residuals against the standardized predicted values (e.g. Figures 7.5) show a random scatter of points with no discernible pattern identified.

- *Independence of the error term.* We tested this assumption with a Durbin-Watson test that did not reveal a significant correlation between residuals
(see Table 7.21). This statistic can vary between 0 and 4, with values between 1.5 and 2.5 indicating that residuals are not correlated seriously.

- Normality of the error term distribution. The results of the Shapiro-Wilk test show that there is a deviation from normality (see 7.22). Nevertheless, the model is robust with respect to this assumption, thus we can accept conclusions based on this model.

Finally, the remaining measures are not included in the models because they

<table>
<thead>
<tr>
<th></th>
<th>Y = OFC₁</th>
<th>Y = OFC₂</th>
<th>Y = OFC₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>Durbin-Watson Statistic</td>
<td>1.518</td>
<td>1.575</td>
<td>1.518</td>
</tr>
</tbody>
</table>

Table 7.21: Independence of the residuals (OFC)
7.5. ANALYSIS AND INTERPRETATION OF THE RESULTS

<table>
<thead>
<tr>
<th></th>
<th>$Y = \text{OFC}_1$</th>
<th>$Y = \text{OFC}_2$</th>
<th>$Y = \text{OFC}_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statistic</td>
<td>0.849</td>
<td>0.832</td>
<td>0.849</td>
</tr>
<tr>
<td>Significance</td>
<td>0.004</td>
<td>0.002</td>
<td>0.004</td>
</tr>
</tbody>
</table>

Table 7.22: Normality of the residuals (OFC): Shapiro-Wilk Test

present multi-collinearity problems.

7.5.5 Threats

Four different threats to the validity of the experiment should be addressed [Wohlin et al., 2000]:

- **Conclusion validity.** An issue that could affect the statistical validity of this study is the size of the sample data, which may not be large enough for a conclusive statistical analysis. Thus, we do not consider these results to be final.

- **Construct validity.** The study was carefully designed, and the design was piloted several times before actually being run. The quality measures are times (TBT), simple counts (TNF) and percentages (OFC), which can be objectively and reliably measured. The behavioural measures used in this study were shown to adequately quantify the attribute they purport to measure in Chapter 5.

- **Internal validity.** The experiments were highly controlled and monitored, so it is very unlikely that undetected influences have occurred without our knowledge. The consistency among observed data sets obtained independently also supports this. The instrumentation was trustworthy since the data was collected, and the measures computed, electronically.

- **External validity.** Although the study is based on a real case, more studies are needed using systems from different enterprise domains. We are aware that more experiments with different platforms (e.g. computer and network hardware, operating systems, etc.) and infrastructure (e.g. middleware type, programming language) must also be carried out to further generalise these
results. As a first step in this direction, we replicated the experiments twice with different platforms, the results of which are presented in section 7.6.

7.6 Replication of Experiments

As a first step to generalise the results obtained in the previous sections, we replicated these experiments running the system over different platforms. In this section, we report the analysis of the results only, since their definition, planning and operation remain identical, except for the following:

- The platform of the computer network was different: the Linux operating system was changed to the Windows operating system in one case, and in the other case the network Ethernet links were replaced by wireless links.

- Each replication was run three (instead of five) different times to check the consistency of the data, and only one (instead of three) data set is reported for the same reason—i.e. to avoid repetition.

7.6.1 First Experiment Replications

7.6.1.1 Descriptive statistics

Table 7.23 presents the descriptive statistics of TBT (in bold). A comparison with the original experiment shows that they are similar in the case of the first replication, while they exhibit higher values in the case of the second replication due to the impact of a slower network. In spite of this, the distribution of the data is similar to the original experiment.

The histogram plots suggested that their distributions were not normal either. This was confirmed by the Shapiro-Wilk test. The results show clearly that the distributions also deviated significantly from normality (see Table 7.24)

7.6.1.2 Correlation Analysis

Table 7.25 presents the correlation coefficients (at the 0.01 significance level) between the behavioural measures and efficiency—measured by TBT. The results
show that the correlation coefficients of the replications are very similar to the original experiment. Therefore, we conclude that, in principle, the results of the correlation analysis seem to be independent of the DIS platform.

### 7.6.1.3 Simple Regression Analysis

Table 7.26 presents the computed regression coefficients ($\beta$), the actual statistical significance of $\beta$, and indicators ($R$, $R^2$ and adjusted $R^2$) of goodness-of-fit of the models. Each row contains the parameter estimates of a simple regression model.

It can be seen that the model goodness-of-fit of the replications are very similar to the original experiment. Consequently, we argue that the results of the simple regression analysis are likely to be independent of the DIS platform.

### 7.6.1.4 Multiple Regression Analysis

Table 7.27 provides the regression coefficients, their statistical significance, and indicators of goodness-of-fit of the models, after performing stepwise multiple linear regression.

The results show that the model goodness-of-fit of the replications are very similar to the original experiment. Hence we conclude that, in principle, the results of the multiple regression analysis seem to be independent of the DIS platform.
<table>
<thead>
<tr>
<th></th>
<th>TBT₁</th>
<th>TBT₅</th>
<th>TBT₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>COS</td>
<td>0.90</td>
<td>0.91</td>
<td>0.90</td>
</tr>
<tr>
<td>COA</td>
<td>0.90</td>
<td>0.91</td>
<td>0.90</td>
</tr>
<tr>
<td>COB</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>CIS</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>CIA</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>CIB</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>IOS</td>
<td>0.94</td>
<td>0.95</td>
<td>0.94</td>
</tr>
<tr>
<td>IOA</td>
<td>0.94</td>
<td>0.95</td>
<td>0.94</td>
</tr>
<tr>
<td>IOB</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>IIS</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>IIA</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>IIB</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>SOS</td>
<td>0.84</td>
<td>0.84</td>
<td>0.83</td>
</tr>
<tr>
<td>SOA</td>
<td>0.84</td>
<td>0.84</td>
<td>0.83</td>
</tr>
<tr>
<td>SOB</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>SIS</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>SIA</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>SIB</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>SL</td>
<td>0.81</td>
<td>0.83</td>
<td>0.82</td>
</tr>
<tr>
<td>AL</td>
<td>0.81</td>
<td>0.83</td>
<td>0.82</td>
</tr>
<tr>
<td>BL</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

Table 7.25: Behavioural measures and TBT: Spearman’s Correlation coefficients (Replications)
<table>
<thead>
<tr>
<th>Y</th>
<th>X</th>
<th>$\beta_0$</th>
<th>$\beta_1$</th>
<th>Sig.</th>
<th>R</th>
<th>$R^2$</th>
<th>Adj. $R^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBT$_4$</td>
<td>COS</td>
<td>0.01</td>
<td>3.25</td>
<td>0.00</td>
<td>0.90</td>
<td>0.81</td>
<td>0.80</td>
</tr>
<tr>
<td>TBT$_5$</td>
<td>COS</td>
<td>0.01</td>
<td>4.39</td>
<td>0.00</td>
<td>0.88</td>
<td>0.78</td>
<td>0.77</td>
</tr>
<tr>
<td>TBT$_2$</td>
<td>COS</td>
<td>0.04</td>
<td>3.01</td>
<td>0.00</td>
<td>0.91</td>
<td>0.82</td>
<td>0.81</td>
</tr>
<tr>
<td>TBT$_4$</td>
<td>COA</td>
<td>0.01</td>
<td>3.25</td>
<td>0.00</td>
<td>0.90</td>
<td>0.81</td>
<td>0.80</td>
</tr>
<tr>
<td>TBT$_5$</td>
<td>COA</td>
<td>0.01</td>
<td>4.39</td>
<td>0.00</td>
<td>0.88</td>
<td>0.78</td>
<td>0.77</td>
</tr>
<tr>
<td>TBT$_2$</td>
<td>COA</td>
<td>0.04</td>
<td>3.01</td>
<td>0.00</td>
<td>0.91</td>
<td>0.82</td>
<td>0.81</td>
</tr>
<tr>
<td>TBT$_4$</td>
<td>IOS</td>
<td>1.17</td>
<td>0.48</td>
<td>0.00</td>
<td>0.86</td>
<td>0.73</td>
<td>0.72</td>
</tr>
<tr>
<td>TBT$_5$</td>
<td>IOS</td>
<td>1.48</td>
<td>0.66</td>
<td>0.00</td>
<td>0.86</td>
<td>0.74</td>
<td>0.73</td>
</tr>
<tr>
<td>TBT$_2$</td>
<td>IOS</td>
<td>1.14</td>
<td>0.44</td>
<td>0.00</td>
<td>0.86</td>
<td>0.74</td>
<td>0.72</td>
</tr>
<tr>
<td>TBT$_4$</td>
<td>IOA</td>
<td>1.17</td>
<td>0.48</td>
<td>0.00</td>
<td>0.86</td>
<td>0.73</td>
<td>0.72</td>
</tr>
<tr>
<td>TBT$_5$</td>
<td>IOA</td>
<td>1.48</td>
<td>0.66</td>
<td>0.00</td>
<td>0.86</td>
<td>0.74</td>
<td>0.73</td>
</tr>
<tr>
<td>TBT$_2$</td>
<td>IOA</td>
<td>1.14</td>
<td>0.44</td>
<td>0.00</td>
<td>0.86</td>
<td>0.74</td>
<td>0.72</td>
</tr>
<tr>
<td>TBT$_4$</td>
<td>SOS</td>
<td>1.17</td>
<td>0.62</td>
<td>0.00</td>
<td>0.76</td>
<td>0.58</td>
<td>0.55</td>
</tr>
<tr>
<td>TBT$_5$</td>
<td>SOS</td>
<td>1.52</td>
<td>0.86</td>
<td>0.00</td>
<td>0.76</td>
<td>0.57</td>
<td>0.55</td>
</tr>
<tr>
<td>TBT$_2$</td>
<td>SOS</td>
<td>1.13</td>
<td>0.58</td>
<td>0.00</td>
<td>0.76</td>
<td>0.58</td>
<td>0.56</td>
</tr>
<tr>
<td>TBT$_4$</td>
<td>SOA</td>
<td>1.17</td>
<td>0.62</td>
<td>0.00</td>
<td>0.76</td>
<td>0.58</td>
<td>0.55</td>
</tr>
<tr>
<td>TBT$_5$</td>
<td>SOA</td>
<td>1.52</td>
<td>0.86</td>
<td>0.00</td>
<td>0.76</td>
<td>0.57</td>
<td>0.55</td>
</tr>
<tr>
<td>TBT$_2$</td>
<td>SOA</td>
<td>1.13</td>
<td>0.58</td>
<td>0.00</td>
<td>0.76</td>
<td>0.58</td>
<td>0.56</td>
</tr>
<tr>
<td>TBT$_4$</td>
<td>SL</td>
<td>0.52</td>
<td>2.56</td>
<td>0.00</td>
<td>0.72</td>
<td>0.51</td>
<td>0.49</td>
</tr>
<tr>
<td>TBT$_5$</td>
<td>SL</td>
<td>0.55</td>
<td>3.56</td>
<td>0.00</td>
<td>0.73</td>
<td>0.53</td>
<td>0.50</td>
</tr>
<tr>
<td>TBT$_2$</td>
<td>SL</td>
<td>0.45</td>
<td>2.42</td>
<td>0.00</td>
<td>0.74</td>
<td>0.55</td>
<td>0.52</td>
</tr>
<tr>
<td>TBT$_4$</td>
<td>AL</td>
<td>0.52</td>
<td>2.56</td>
<td>0.00</td>
<td>0.72</td>
<td>0.51</td>
<td>0.49</td>
</tr>
<tr>
<td>TBT$_5$</td>
<td>AL</td>
<td>0.55</td>
<td>3.56</td>
<td>0.00</td>
<td>0.73</td>
<td>0.53</td>
<td>0.50</td>
</tr>
<tr>
<td>TBT$_2$</td>
<td>AL</td>
<td>0.45</td>
<td>2.42</td>
<td>0.00</td>
<td>0.74</td>
<td>0.55</td>
<td>0.52</td>
</tr>
</tbody>
</table>

Table 7.26: Behavioural Measures and TBT: Simple Regression Statistics (Replications)
<table>
<thead>
<tr>
<th>$X_i$</th>
<th>$\beta_i$</th>
<th>Sig.</th>
<th>$Y$</th>
<th>$R$</th>
<th>$R^2$</th>
<th>Adj. $R^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>constant</td>
<td>-0.13</td>
<td>&gt; 0.05</td>
<td>TBT_4</td>
<td>0.92</td>
<td>0.85</td>
<td>0.83</td>
</tr>
<tr>
<td>IOS</td>
<td>0.37</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL</td>
<td>1.38</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>constant</td>
<td>-0.34</td>
<td>&gt; 0.05</td>
<td>TBT_5</td>
<td>0.93</td>
<td>0.86</td>
<td>0.85</td>
</tr>
<tr>
<td>IOS</td>
<td>0.51</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL</td>
<td>1.94</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>constant</td>
<td>-0.13</td>
<td>&gt; 0.05</td>
<td>TBT_2</td>
<td>0.93</td>
<td>0.87</td>
<td>0.85</td>
</tr>
<tr>
<td>IOS</td>
<td>0.33</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL</td>
<td>1.36</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>constant</td>
<td>-0.32</td>
<td>&gt; 0.05</td>
<td>TBT_4</td>
<td>0.86</td>
<td>0.74</td>
<td>0.71</td>
</tr>
<tr>
<td>SOA</td>
<td>1.64</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AL</td>
<td>0.45</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>constant</td>
<td>-0.58</td>
<td>&gt; 0.05</td>
<td>TBT_5</td>
<td>0.86</td>
<td>0.75</td>
<td>0.72</td>
</tr>
<tr>
<td>SOA</td>
<td>0.60</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AL</td>
<td>2.32</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>constant</td>
<td>-0.31</td>
<td>&gt; 0.05</td>
<td>TBT_2</td>
<td>0.88</td>
<td>0.77</td>
<td>0.74</td>
</tr>
<tr>
<td>SOA</td>
<td>0.40</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AL</td>
<td>1.59</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7.27: Behavioural Measures and TBT: Multiple Regression Statistics (Replications)
7.6. REPLICAION OF EXPERIMENTS

<table>
<thead>
<tr>
<th>Measure</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Sum</th>
<th>Mean</th>
<th>Std. Dev.</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>TNF₄</td>
<td>0.00</td>
<td>12.48</td>
<td>95.36</td>
<td>4.54</td>
<td>4.79</td>
<td>23.02</td>
</tr>
<tr>
<td>TNF₅</td>
<td>0.00</td>
<td>12.56</td>
<td>94.43</td>
<td>4.50</td>
<td>4.77</td>
<td>22.74</td>
</tr>
<tr>
<td>TNF₆</td>
<td>0.00</td>
<td>12.11</td>
<td>88.84</td>
<td>4.23</td>
<td>4.54</td>
<td>20.57</td>
</tr>
</tbody>
</table>

Table 7.28: TNF: Descriptive Statistics (n = 21)

<table>
<thead>
<tr>
<th>Measure</th>
<th>Shapiro-Wilk Statistic</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>TNF₄</td>
<td>0.823</td>
<td>0.002</td>
</tr>
<tr>
<td>TNF₅</td>
<td>0.821</td>
<td>0.001</td>
</tr>
<tr>
<td>TNF₆</td>
<td>0.825</td>
<td>0.002</td>
</tr>
</tbody>
</table>

Table 7.29: TNF: Normality Test (n = 21)

7.6.2 Second Experiment Replications

7.6.2.1 Descriptive statistics

Table 7.28 presents the descriptive statistics of TNF (in bold). A comparison with the original experiment shows that the values are similar to both replications. Furthermore, the distributions of the data are similar to the original experiment.

The histogram plots of the measures suggested that their distributions were not normal either. This was confirmed by the Shapiro-Wilk test (see Table 7.29). It can be seen that their distribution also deviated significantly from normality.

7.6.2.2 Correlation Analysis

Table 7.30 presents the correlation coefficients (at the 0.01 significance level) between the behavioural measures and reliability—measured by TNF. The results show that the correlations of the replications are very similar to the original experiment. Therefore, we conclude that, in principle, the results of the correlation analysis seem to be independent of the DiS platform.
<table>
<thead>
<tr>
<th></th>
<th>$TBT_4$</th>
<th>$TBT_5$</th>
<th>$TBT_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>COS</td>
<td>0.93</td>
<td>0.92</td>
<td>0.90</td>
</tr>
<tr>
<td>COA</td>
<td>0.93</td>
<td>0.92</td>
<td>0.90</td>
</tr>
<tr>
<td>COB</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>CIS</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>CIA</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>CIB</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>IOS</td>
<td>0.91</td>
<td>0.92</td>
<td>0.94</td>
</tr>
<tr>
<td>IOA</td>
<td>0.91</td>
<td>0.92</td>
<td>0.94</td>
</tr>
<tr>
<td>IOB</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>IIS</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>IIA</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>IIB</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>SOS</td>
<td>0.86</td>
<td>0.86</td>
<td>0.83</td>
</tr>
<tr>
<td>SOA</td>
<td>0.86</td>
<td>0.86</td>
<td>0.83</td>
</tr>
<tr>
<td>SOB</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>SIS</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>SIA</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>SIB</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>SL</td>
<td>0.80</td>
<td>0.79</td>
<td>0.82</td>
</tr>
<tr>
<td>AL</td>
<td>0.80</td>
<td>0.79</td>
<td>0.82</td>
</tr>
<tr>
<td>BL</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

Table 7.30: Behavioural measures and TNF: Spearman's correlation coefficients (Replications)
7.6.2.3 Simple Regression Analysis

Table 7.31 presents the computed regression coefficients ($\beta$), the actual statistical significance of $\beta$, and indicators ($R$, $R^2$ and adjusted $R^2$) of goodness-of-fit of the models. Each row contains the parameter estimates of a simple regression model.

It can be seen that the model goodness-of-fit of the replications are very similar to the original experiment. Consequently, we argue that the results of the simple regression analysis are likely to be independent of the DIS platform.

7.6.2.4 Multiple Regression Analysis

Table 7.32 provides the regression coefficients, their statistical significance, and indicators of goodness-of-fit of the models, after performing stepwise multiple linear regression.

The results show that the model goodness-of-fit of the replications are very similar to the original experiment. Hence we conclude that, in principle, the results of the multiple regression analysis seem to be independent of the DIS platform.

7.7 Discussion

The first objective of the experiments was to test whether the behavioural measures were correlated to the quality attributes of interest. The overall conclusion is that a significant number of the measures are highly correlated to quality attributes. It should be noted that some of the measures could not be tested due to the unsuitability of the values obtained for those measures—all values equal to zero. Table 7.33 summarises the relationships among behavioural measures and quality attributes where:

- 'Y' indicates the relationship was supported by our empirical data,
- 'N' implies the relationship was not supported by our empirical data, and
- '?' indicates the relationship could not be tested with our empirical data.

The second objective was to test whether the behavioural measures can be useful individual estimators of quality attributes. The overall conclusion is that all
<table>
<thead>
<tr>
<th>$Y$</th>
<th>$X$</th>
<th>$\hat{\beta}_0$</th>
<th>$\hat{\beta}_1$</th>
<th>Sig.</th>
<th>$R$</th>
<th>$R^2$</th>
<th>Adj. $R^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>TNF$_4$</td>
<td>COS</td>
<td>-0.12</td>
<td>3.64</td>
<td>0.00</td>
<td>0.91</td>
<td>0.82</td>
<td>0.81</td>
</tr>
<tr>
<td>TNF$_5$</td>
<td>COS</td>
<td>-0.15</td>
<td>3.61</td>
<td>0.00</td>
<td>0.90</td>
<td>0.81</td>
<td>0.80</td>
</tr>
<tr>
<td>TNF$_2$</td>
<td>COS</td>
<td>0.04</td>
<td>3.01</td>
<td>0.00</td>
<td>0.91</td>
<td>0.82</td>
<td>0.81</td>
</tr>
<tr>
<td>TNF$_4$</td>
<td>COA</td>
<td>-0.12</td>
<td>3.64</td>
<td>0.00</td>
<td>0.91</td>
<td>0.82</td>
<td>0.81</td>
</tr>
<tr>
<td>TNF$_5$</td>
<td>COA</td>
<td>-0.15</td>
<td>3.61</td>
<td>0.00</td>
<td>0.90</td>
<td>0.81</td>
<td>0.80</td>
</tr>
<tr>
<td>TNF$_2$</td>
<td>COA</td>
<td>0.04</td>
<td>3.01</td>
<td>0.00</td>
<td>0.91</td>
<td>0.82</td>
<td>0.81</td>
</tr>
<tr>
<td>TNF$_4$</td>
<td>IOS</td>
<td>1.26</td>
<td>0.53</td>
<td>0.00</td>
<td>0.85</td>
<td>0.72</td>
<td>0.71</td>
</tr>
<tr>
<td>TNF$_5$</td>
<td>IOS</td>
<td>1.15</td>
<td>0.53</td>
<td>0.00</td>
<td>0.86</td>
<td>0.74</td>
<td>0.73</td>
</tr>
<tr>
<td>TNF$_2$</td>
<td>IOS</td>
<td>1.14</td>
<td>0.44</td>
<td>0.00</td>
<td>0.86</td>
<td>0.74</td>
<td>0.72</td>
</tr>
<tr>
<td>TNF$_4$</td>
<td>IOA</td>
<td>1.26</td>
<td>0.53</td>
<td>0.00</td>
<td>0.85</td>
<td>0.72</td>
<td>0.71</td>
</tr>
<tr>
<td>TNF$_5$</td>
<td>IOA</td>
<td>1.15</td>
<td>0.53</td>
<td>0.00</td>
<td>0.86</td>
<td>0.74</td>
<td>0.73</td>
</tr>
<tr>
<td>TNF$_2$</td>
<td>IOA</td>
<td>1.14</td>
<td>0.44</td>
<td>0.00</td>
<td>0.86</td>
<td>0.74</td>
<td>0.72</td>
</tr>
<tr>
<td>TNF$_4$</td>
<td>SOS</td>
<td>1.18</td>
<td>0.70</td>
<td>0.00</td>
<td>0.77</td>
<td>0.59</td>
<td>0.57</td>
</tr>
<tr>
<td>TNF$_5$</td>
<td>SOS</td>
<td>1.07</td>
<td>0.71</td>
<td>0.00</td>
<td>0.78</td>
<td>0.61</td>
<td>0.59</td>
</tr>
<tr>
<td>TNF$_2$</td>
<td>SOS</td>
<td>1.13</td>
<td>0.58</td>
<td>0.00</td>
<td>0.76</td>
<td>0.58</td>
<td>0.56</td>
</tr>
<tr>
<td>TNF$_4$</td>
<td>SOA</td>
<td>1.18</td>
<td>0.70</td>
<td>0.00</td>
<td>0.77</td>
<td>0.59</td>
<td>0.57</td>
</tr>
<tr>
<td>TNF$_5$</td>
<td>SOA</td>
<td>1.07</td>
<td>0.71</td>
<td>0.00</td>
<td>0.78</td>
<td>0.61</td>
<td>0.59</td>
</tr>
<tr>
<td>TNF$_2$</td>
<td>SOA</td>
<td>1.13</td>
<td>0.58</td>
<td>0.00</td>
<td>0.76</td>
<td>0.58</td>
<td>0.56</td>
</tr>
<tr>
<td>TNF$_4$</td>
<td>SL</td>
<td>0.25</td>
<td>3.02</td>
<td>0.00</td>
<td>0.76</td>
<td>0.58</td>
<td>0.56</td>
</tr>
<tr>
<td>TNF$_5$</td>
<td>SL</td>
<td>0.36</td>
<td>2.90</td>
<td>0.00</td>
<td>0.73</td>
<td>0.54</td>
<td>0.51</td>
</tr>
<tr>
<td>TNF$_2$</td>
<td>SL</td>
<td>0.45</td>
<td>2.42</td>
<td>0.00</td>
<td>0.74</td>
<td>0.55</td>
<td>0.52</td>
</tr>
<tr>
<td>TNF$_4$</td>
<td>AL</td>
<td>0.25</td>
<td>3.02</td>
<td>0.00</td>
<td>0.76</td>
<td>0.58</td>
<td>0.56</td>
</tr>
<tr>
<td>TNF$_5$</td>
<td>AL</td>
<td>0.36</td>
<td>2.90</td>
<td>0.00</td>
<td>0.73</td>
<td>0.54</td>
<td>0.51</td>
</tr>
<tr>
<td>TNF$_2$</td>
<td>AL</td>
<td>0.45</td>
<td>2.42</td>
<td>0.00</td>
<td>0.74</td>
<td>0.55</td>
<td>0.52</td>
</tr>
</tbody>
</table>

Table 7.31: Behavioural Measures and TNF: Simple Regression Statistics (Replications)
<table>
<thead>
<tr>
<th>$X_i$</th>
<th>$\beta_i$</th>
<th>Sig.</th>
<th>$Y$</th>
<th>$R$</th>
<th>$R^2$</th>
<th>Adj. $R^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>constant</td>
<td>-0.43</td>
<td>&lt; 0.05</td>
<td>TNF$_4$</td>
<td>0.94</td>
<td>0.87</td>
<td>0.86</td>
</tr>
<tr>
<td>IOA</td>
<td>0.39</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AL</td>
<td>1.79</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>constant</td>
<td>-0.36</td>
<td>&lt; 0.05</td>
<td>TNF$_5$</td>
<td>0.93</td>
<td>0.86</td>
<td>0.85</td>
</tr>
<tr>
<td>IOA</td>
<td>0.41</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AL</td>
<td>1.60</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>constant</td>
<td>-0.13</td>
<td>&lt; 0.05</td>
<td>TNF$_2$</td>
<td>0.93</td>
<td>0.87</td>
<td>0.85</td>
</tr>
<tr>
<td>IOA</td>
<td>0.33</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AL</td>
<td>1.36</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>constant</td>
<td>-0.65</td>
<td>&lt; 0.05</td>
<td>TNF$_4$</td>
<td>0.89</td>
<td>0.80</td>
<td>0.77</td>
</tr>
<tr>
<td>SOS</td>
<td>0.48</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL</td>
<td>2.04</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>constant</td>
<td>-0.60</td>
<td>&lt; 0.05</td>
<td>TNF$_5$</td>
<td>0.88</td>
<td>0.78</td>
<td>0.76</td>
</tr>
<tr>
<td>SOS</td>
<td>0.51</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL</td>
<td>1.85</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>constant</td>
<td>-0.31</td>
<td>&lt; 0.05</td>
<td>TNF$_2$</td>
<td>0.88</td>
<td>0.77</td>
<td>0.74</td>
</tr>
<tr>
<td>SOS</td>
<td>0.40</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL</td>
<td>1.59</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7.32: Behavioural Measures and TNF: Multiple Regression Statistics (Replications)
<table>
<thead>
<tr>
<th></th>
<th>Efficiency (TBT)</th>
<th>Reliability (TNF)</th>
<th>Maintainability (OFC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>COS</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>COA</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>COC</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>COB</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>CIS</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>CIA</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>CIC</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>CIB</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>IOS</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>IOA</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>IOC</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>IOB</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>IIS</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>IIA</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>IIC</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>IIB</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>SOS</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>SOA</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>SOC</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>SOB</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>SIS</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>SIA</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>SIC</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>SIB</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>SL</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>AL</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>CL</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>BL</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

Table 7.33: Summary of relationships among behavioural measures and quality attributes
measures correlated to quality attributes are good individual predictors.

The third objective was to test whether individual predictors can be used in combination to estimate quality attributes. We found several statistically significant multiple models with high goodness-of-fit.

These results cannot be considered final, and are only considered preliminary findings. More experiments and replications are necessary to arrive to conclusive results. As a first step in this direction, we replicated the first and the second experiment on different platforms. The changes in the platform (a different operating system and a different network) did not affect the relationships among the behavioural attributes and the quality attributes. Furthermore, in all cases, the results of the replications were highly consistent with the original experiments. This may be taken as a strong indication that the results are independent of the platform.

7.8 Summary

In this chapter, we have empirically tested the proposed behavioural measures as indicators of the quality attributes of interest with data extracted from a DIS. The results suggest that a significant number of these measures are correlated to and can be useful estimators of quality attributes. The practical applicability of the proposed measures is examined in the next chapter.
Chapter 8

Practical Considerations

8.1 Introduction

In this chapter we discuss issues on the practical applicability of the proposed measures. It has been found that if software measures are difficult or labour intensive to collect, software engineers and managers will be reluctant to use them. Section 8.2 describes briefly the design and implementation of an electronic tool for the automatic collection of the proposed software measures, and the estimation of the indicators of quality attributes. The generic abstractions were the foundation to develop an electronic tool that collects the measures and produces quality indicators. The tool was designed to be flexible, to be able to produce information from different inputs, and to produce output in different formats.

Software measures are most useful in practice when they are integrated into the software process [Henderson-Sellers, 1996]. In Section 8.3, we discuss how the software measures—with the tool—may be applied to the software process to help engineers make decisions.

8.2 Automatic tool

The literature indicates that it is highly desirable to demonstrate that is possible to automate the collection of the proposed measures [Card and Glass, 1990; Grady and Caswell, 1987]. Technological support is very important for the effective use
of software measures: Some of the advantages of a tool that can automate the collection of measures are [Lavazza, 2000]:

- the possibility of obtaining measures easily;
- minimising errors in calculating measures, thus obtaining more precise results; and
- the focus is centred on the analysis of the results, not on data extraction.

Consequently, we designed and implemented a prototype to extract measures from software design artefacts and to provide estimates of the quality attributes of interest. This application was implemented in Java and the source code is available from the World Wide Web\(^1\). The tool consists of four main components:

- A component that transforms the input into the internal representation\(^2\) of the generic abstractions. Currently, the application can transform input from comma delimited text files and from relational databases compatible with JDBC.

- A component that computes the values of measures from the internal representation of the abstractions. At present, the tool is configured to collect values for the measures proposed in this dissertation.

- A component that produces estimates for quality attributes. At the moment, the application produces estimates for the quality attributes addressed in this investigation.

- A component that formats and outputs the produced results—i.e. the measures and estimations. The tool presently produces the output as comma delimited text files (suitable for spreadsheet use) and HTML files.

Figure 8.1 shows a generic collaboration diagram of the prototype that illustrates the interaction among application components.

\(^1\)http://www.cs.rmit.edu.au/~pablo/research/phd/

\(^2\)The data structure used to represent the generic abstractions is a set of tuples, which was implemented as a list of arrays.
The application was designed to be flexible and adaptive. It is possible to plug-in new input or output capabilities, as well as new measures or estimates, with minimal or no change at all to the application structure and behaviour.

Finally, it should be noted that the tool was used successfully to collect the proposed measures accurately during the empirical evaluation. Figures 8.2 and 8.3 shows input and output text files, respectively, with sample data from the experimentation.

### 8.3 Product Measurement and the Production Process

Chapters 6 and 7 demonstrated that the measures could be used for the estimation of quality attributes. In this section we discuss how these measures can help the practitioner to make decisions in the software development process of DIS.

During or after the design of DIS, we believe most software engineers would be interested in obtaining answers to questions such as:

- Which components of the system are more failure prone?
#component, cluster, dependency_type, component, cluster
ui, sa, p, cs, sa
mr, sa, p, cs, sa
cs, sa, p, da, sa
cs, sa, p, ms, sa
ms, sa, i, mr, ri
ui, ca, p, cs, ca
mr, ca, p, cs, ca
cs, ca, p, da, ca
cs, ca, p, ms, ca
ms, ca, i, mr, ri
...

Figure 8.2: Tool: Sample Input File

CLUSTER, CLOP, CLOI, CLIP, CLII, COOP, COOI, COIP, COII, DOP, DOI, DIP, DII
cacs, 3, 0, 2, 0, 1, 0, 1, 0, 3, 0, 2, 0
csda, 2, 0, 1, 0, 1, 0, 1, 0, 2, 0, 1, 0
caddb, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0
caimr, 2, 0, 0, 0, 1, 1, 0, 0, 1, 2, 0, 0, 1
cams, 1, 2, 1, 0, 1, 1, 1, 0, 1, 2, 1, 0
cans, 0, 0, 5, 0, 0, 0, 1, 0, 0, 0, 0, 5, 0
cau, 2, 0, 0, 0, 1, 0, 0, 0, 2, 0, 0, 0, 0
rics, 3, 0, 2, 0, 1, 0, 1, 0, 3, 0, 2, 0
rida, 2, 0, 1, 0, 1, 0, 1, 0, 2, 0, 1, 0
ridd, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0
rimr, 2, 0, 0, 2, 1, 0, 0, 1, 2, 0, 0, 0, 0
rims, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0
rins, 0, 0, 5, 0, 0, 0, 1, 0, 0, 0, 0, 5, 0
riui, 2, 0, 0, 0, 1, 0, 0, 0, 2, 0, 0, 0, 0
sacs, 3, 0, 2, 0, 1, 0, 1, 0, 3, 0, 2, 0
sada, 2, 0, 1, 0, 1, 0, 1, 0, 2, 0, 1, 0
sadb, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0
samr, 2, 0, 0, 2, 1, 0, 0, 1, 2, 0, 0, 0, 0
sams, 1, 2, 1, 0, 1, 1, 1, 0, 1, 2, 1, 0
sans, 0, 0, 5, 0, 0, 0, 1, 0, 0, 0, 0, 5, 0
saui, 2, 0, 0, 0, 1, 0, 0, 0, 2, 0, 0, 0, 0
...

Figure 8.3: Tool: Sample Output File
8.3. PRODUCT MEASUREMENT AND THE PRODUCTION PROCESS

- Which components of the system are most likely to be less efficient?
- Which components can be modified without affecting the rest of the system?
- What is the best strategy to cluster system components?
- How much testing should be planned for a cluster of components?
- What are the best options to allocate and deploy clusters of components to the available computer and network hardware?

We argue that the proposed measures provide information that may help software engineers to answer these questions and make more objective decisions.

A decision that is particularly unique to the development of DIS is the allocation of software components to network nodes, and in some cases the allocation of computers to sub-networks. The measures COS and COA may be used to identify strongly coupled components to cluster and deploy them to the same sub-network. Testing and maintenance of inter-components interactions will be easier generally if the components are close physically. A cluster of highly interrelated components may be allocated to high-speed sub-networks and/or to powerful computers to alleviate potential efficiency problems.

As discussed previously, enterprise DIS are not always designed from scratch. That is, often the design of an enterprise DIS involves the integration of new components and legacy components. In these cases, the measures CLOP and CLIP may be helpful to compare different boundaries to (re)cluster legacy and new components, following one of the key principles that is to minimise inter-cluster (processing) dependencies.

Another decision that is distinctive of the development of DIS is the selection of middleware interfaces. We argued that a procedure (or function call) is not the only way for distributed components to interact. Several mechanisms exist that exhibit different levels of coupling, in a distributed computing sense. In this case, the measures CLOI and CLI can be used to identify mismatches between the interfaces of the supporting middleware and the required structure and behaviour of the system components.
The measures DOP and IOS can be used to check the consistency between the structure and behaviour of the design. For example, if a cluster is composed of components with no inter-cluster processing dependencies and some of these components collaborate synchronously in their cooperative scenarios, then a potential opportunity for improvement is identified. Components can interact asynchronously if their dependency is informational, and improve their extrinsic quality. On the other hand, if a cluster is composed of components that exhibit inter-cluster processing dependencies, but none of these components collaborate synchronously in their cooperative scenarios, there is an indication of likely integrity problems since usually processing dependencies cannot be implemented asynchronously.

During operation, the measures IIA and CIA may be useful to identify which components can be taken off-line, to be upgraded or tested without disrupting the rest of the cluster or system. For instance, components that are not involved in available interactions—IIA = 0 or CIA = 0—can be temporarily shut down without disturbing the operation of other components.

In the case of reallocation or redistribution of components, the measures SIB and BL can help to evaluate the impact on other components. Components coupled by incoming statically bound interactions will cause problems to other components in most cases.

8.4 Summary

In this chapter we have discussed the issues of the practical applicability of the proposed measures. Section 8.2 described the design and implementation of a electronic tool for the automatic collection of the software measures and the estimation of the indicators of quality attributes. Section 8.3 examined the use of the measures as part of the software development process. We conclude this dissertation in the next chapter with an evaluation of the research objectives and a discussion of future research directions.
Chapter 9

Conclusions

9.1 Introduction

This research was motivated by the recognition that software engineering of distributed applications is an increasingly important activity in enterprises. It was also predicated that well constructed and meaningful measures can help enterprises to understand, evaluate and better use distributed technologies.

However, the majority of the existent literature on software measures is focused on traditional (centralised) information systems. Although there have been some proposals for DIS, all but one are derived from information extracted from code. The agenda for this dissertation was to propose design measures for DIS, which can be used as early indicators of the quality of the executable software.

By developing and evaluating measures with a methodological approach, which are both theoretically sound and practically relevant, this dissertation claims to have addressed the agenda. Furthermore, the ideas and some of the results of this work have been introduced to the research community and have been accepted at international refereed forums [Fernandez and Rossi, 2000; Rossi and Fernandez, 2002, 2003, 2004].
9.2 Evaluation of the Research Objectives and Contributions

In this section, we revisit the main objective of this investigation and show how the dissertation addresses this objective by achieving the sub-objectives identified in Chapter 1.

We stated that the main objective addressed by this dissertation is to define a suite of software design measures to be used as early indicators of quality attributes for enterprise DIS. To evaluate the main objective we have to analyse each of its specific sub-objectives.

Analyse critically the state of the art of measurement for software products, in particular for DIS artefacts. A considerable large amount of software measures can be identified in the literature for traditional (centralised) information systems. However, less research effort has been directed towards DIS software artefacts. Although some proposed measures for DIS exist, all except one address the measurement of attributes of software code. These proposals were examined to determine whether the measures have been defined rigorously and evaluated empirically. Some examples of traditional measures are also reviewed for comparative purposes. In addition we have analysed the distinctive characteristics of DIS to identify key design and quality attributes relevant to this type of software (Chapter 2).

Approach the measurement of software artefacts with a well-defined methodology. A disciplined and systematic procedure is necessary to develop measures rigorously. Using as template one comprehensive procedure available in the literature and several other suggestions, we have outlined a plan that addresses the development of software measures in a methodological manner (Chapter 3).

Define theoretically valid software design measures, based on practical goals and hypotheses that are consistent with the intuitive analysis of the problem domain. We have established the goals and hypotheses that motivate the definition of measures and have defined generic abstractions to characterise design attributes of interest (Chapter 4). These elements of the formal model are originated from the intuitive model of the problem domain. The measures were defined following a rigorous
framework that constructs measures systematically that are theoretically valid (ac-
ccording to the prescriptions of the theory of measurement) and measure what they
intend to measure (Chapter 5).

_Evaluate the empirical validity and practical usefulness of the measures as indi-
cators of quality attributes_. Several experiments and replications have been carried
out in order to assess the capability of the measures to be used as indicators of
some quality attributes (Chapters 6 and 7). The majority of the measures are sig-
nificantly and strongly correlated with the quality attributes of interest. Several
exploratory regression models with significant high goodness-of-fit show that most
of the correlated measures can be used individually or in combination for early
estimation of quality attributes. The behaviourial measures show slightly better
individual and combined results. This is in line with our assumptions and ex-
pectations, since they are evaluated against non-functional quality attributes of
the operational system. Nevertheless, the structural measures also provide useful
feedback and are generally available before the behaviourial measures. The results
of the experiments and the replications seem to indicate that the conclusions are
consistent across different platforms—i.e. operating system and network.

_Design and implement a tool for the automatic collection of the software mea-
sures_. If software measures are difficult or labour intensive to collect, software
engineers and managers will be reluctant to use them. Based on generic abstrac-
tions, an electronic tool for the automatic collection the measures was designed and
implemented, which also produces indicators of the software quality attributes of
interest. The tool was designed to be flexible and adaptive since it can produce
information in different formats from different inputs, which are transformed into
suitable internal representations of the generic abstractions for processing (Chap-
ter 8).

_Examine the use of the measures as part of the software development process_.
For software measures to be useful in practice, they have to be integrated to the
software engineering process. We have discussed how the software measures can
be used in the software process by engineers to assess the impact of their decisions
on quality attributes (Chapter 8).

From the above discussion we conclude that, since the sub-objectives have been
achieved, their main objective of the dissertation has been accomplished.

9.3 Future Research Directions

The work of this dissertation has opened a gamut of new research issues. The scope of this project compels us to leave these for future work:

- In order to extend the generality of the obtained results, more replications of the experiments and new experiments are necessary. On the one hand, the empirical evaluation suggests that the results are, in principle, independent of the platform—i.e. network type and operating system. However, we would like to replicate the experiments with a different infrastructure (i.e. middleware) such as Web Service and .NET, and with a different implementation of the system in a language such as C++ or C#. On the other hand, it is also necessary to experiment with a larger system, preferably from a different type of enterprise.

- Another important area for future research is the measurement of other attributes and sub-attributes of design artefacts, and their relationships with attributes of quality, to improve our understanding of this type of software. For instance, we believe that some coupling sub-attributes may be related to the stability and adaptability of DIS components.

- Given that most attributes are composed of several sub-attributes, it is also important to investigate how sub-attribute indicators can be combined to provide an overall indicator for a quality attribute. At least two types of models can be explored: those based on linear additive scoring models [Gilb, 1976] and those based on nonlinear multi-criteria scoring models [Olsina and Rossi, 2002] where different attributes and sub-attributes relationships can be designed.

- We believe the estimation of quality attributes can be improved by including parameters that feed the models with information about the environment. For example, some index of network speed and probabilities of execution could be useful to improve the behavioural models of efficiency.
Finally, due to the increasing capability and availability of smart phones, personal digital assistants (PDA) and laptops, we would like to extend measurement to distributed systems where the software and/or the hardware are mobile.
Appendix A

Experimental Material – Structural Measures

In this appendix we present raw data obtained during the experiments described in Chapter 6.
Table A.1: Values of the Structural Measures (Component and Cluster Perspective)

<table>
<thead>
<tr>
<th></th>
<th>CLOP</th>
<th>CLOI</th>
<th>CLIP</th>
<th>CLI1</th>
<th>COOP</th>
<th>COOI</th>
<th>COIP</th>
<th>COII</th>
</tr>
</thead>
<tbody>
<tr>
<td>cacs</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>csda</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>cadb</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>camr</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>cams</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>cans</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>caui</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>rics</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rida</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rdib</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rimr</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>rims</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rinsi</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>riui</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sacs</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>sada</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>sadb</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>samr</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>sams</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>sans</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>saui</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>DOP</td>
<td>DOI</td>
<td>DIP</td>
<td>DII</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cacs</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>csda</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cadb</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>camr</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cams</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cans</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>caui</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rics</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rida</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ridb</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rimr</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rims</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rins</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>riui</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sacs</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sada</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sadb</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>samr</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sams</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sans</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>saui</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table A.2: Values of the Structural Measures (Dependency Perspective)
Figure A.1: Enterprise DIS structure
Appendix B

Experimental Material – Behavioural Measures

In this appendix we present raw data obtained during the experiments described in Chapter 7.
<table>
<thead>
<tr>
<th></th>
<th>COS</th>
<th>COA</th>
<th>COC</th>
<th>COB</th>
<th>CIS</th>
<th>CIA</th>
<th>CIC</th>
<th>CIB</th>
</tr>
</thead>
<tbody>
<tr>
<td>cacs</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>csda</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>cadb</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>camr</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>cams</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>cans</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>caui</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>rics</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>rida</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ridb</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>rimr</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>rims</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>rins</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ritu</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sacs</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sada</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sadb</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>samr</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>sams</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sans</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>saui</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table B.1: Values of the Behavioural Measures (Component Perspective)
<table>
<thead>
<tr>
<th></th>
<th>IOS</th>
<th>IOA</th>
<th>IOC</th>
<th>IOB</th>
<th>IIS</th>
<th>IIA</th>
<th>IIC</th>
<th>IIB</th>
</tr>
</thead>
<tbody>
<tr>
<td>cacs</td>
<td>19</td>
<td>19</td>
<td>0</td>
<td>0</td>
<td>11</td>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>csda</td>
<td>7</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>16</td>
<td>16</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>cdab</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>camr</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>cams</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>cans</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>caui</td>
<td>9</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>rics</td>
<td>28</td>
<td>28</td>
<td>0</td>
<td>0</td>
<td>14</td>
<td>14</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>rida</td>
<td>7</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>26</td>
<td>26</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ridd</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>rimr</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>rims</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>rins</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>rui</td>
<td>11</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sacs</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sada</td>
<td>7</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>16</td>
<td>16</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sadb</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>samr</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>sams</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sans</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>saui</td>
<td>8</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table B.2: Values of the Behavioural Measures (Interaction perspective)
<table>
<thead>
<tr>
<th></th>
<th>SOS</th>
<th>SOA</th>
<th>SOC</th>
<th>SOB</th>
<th>SIS</th>
<th>SIA</th>
<th>SIC</th>
<th>SIB</th>
</tr>
</thead>
<tbody>
<tr>
<td>cacs</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>csda</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>cadb</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>camr</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>cams</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>cans</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>caui</td>
<td>8</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>rics</td>
<td>15</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>15</td>
<td>15</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>rida</td>
<td>15</td>
<td>15</td>
<td>0</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ridb</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>rimr</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>rims</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>rins</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>riui</td>
<td>11</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sacs</td>
<td>8</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sada</td>
<td>8</td>
<td>8</td>
<td>0</td>
<td>15</td>
<td>8</td>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sadb</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>8</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>samr</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>sams</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sans</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>saui</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table B.3: Values of the Behavioural Measures (Scenario Perspective)
<table>
<thead>
<tr>
<th></th>
<th>SL</th>
<th>AL</th>
<th>CL</th>
<th>BL</th>
</tr>
</thead>
<tbody>
<tr>
<td>cacs</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>csda</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>cadb</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>camr</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>cams</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>cans</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>caui</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>rics</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>rida</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>ridb</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>rimr</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>rims</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>rins</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>riui</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sacs</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sada</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sadb</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>samr</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sams</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>sans</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>saui</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table B.4: Values of the Behavioural Measures (Indirect perspective)
Bibliography


