A SPECIALISED ARCHITECTURE FOR EMBEDDING TRUST EVALUATION CAPABILITIES IN INTELLIGENT MOBILE AGENTS

by

JUSTIN RICHARD PIKE

DISSERTATION

submitted in the fulfilment
of the requirements for the degree

MASTER OF SCIENCE

in COMPUTER SCIENCE

in the

FACULTY OF SCIENCE

at the

UNIVERSITY OF JOHANNESBURG

SUPERVISOR: PROF E.M. EHLERS
CO SUPERVISOR: MR O.L. OOSTHUIZEN

FEBRUARY 2008
Abstract

The dissertation investigates trust and reputation as a specialisation of agent technology. The research presented herein aims to establish and demonstrate how it is possible for one rational agent to trust another entity. Furthermore, the research presented herein aims to determine the extent of the limitations of trust and reputation models, and of the demonstrable solution in particular. To this end, the dissertation investigates theoretical aspects of trust. The dissertation investigates several existing trust models and establishes criteria for a qualitative analysis. Supplementary techniques aimed at enhancing trust evaluation are also investigated. The research also identifies architectural abstractions suitable for developing agents capable of intelligent trust evaluation. The main focus of the research is enhancing agent protection through a trust-based approach. A particular problem is the threats posed to mobile agents from malicious agent hosts. Therefore, a solution is sought that can be used to augment existing mechanisms aimed at mobile agent protection and agent protection in general. Thus, the research also examines mobile agents and mobile agent systems in an effort to produce a general trust-based solution that can be applied in most mobile agent systems. The solution presented in the dissertation proposes the concept of an evaluator agent as an add-on to existing mobile agent systems. The evaluator agent is presented as a rational agent with an embedded intelligent trust evaluation capability. The intelligent trust evaluation capability is provided via a set of reusable components. The solution demonstrates how a rational agent may evaluate the trustworthiness of other entities. The dissertation further analyses the strengths and limitations of the approach. The dissertation provides results that quantitatively demonstrate the extent of the limitations of the trust-based approach. The contribution of the dissertation partly lies in the service orientation of the evaluator agent approach. The service orientation of the solution provides an abstraction and a degree of heterogeneity suitable for handling the challenges of open environments. The solution can be deployed in most mobile agent systems to provide a trust evaluation service without the need to redesign existing mobile agent systems. More broadly, the research is another step towards the development of cognitive social agents.
Keywords

Trust and Reputation, Agents, Agent Architecture, Agent Mobility
Contents

Part I

1. Problem Statement 1
   1.1 Introduction 1
   1.2 The Contextual Background of the Problem 1
      1.2.1 Introduction 1
      1.2.2 Evaluating the Trustworthiness of Agents 2
      1.2.3 Evaluating the Trustworthiness of Agent Hosts 2
      1.2.4 The Need for an Intelligent Approach to Trust 3
      1.2.5 Summary 3
   1.3 The Problem Statement 3
      1.3.1 A Concise Problem Definition 3
      1.3.2 Delineation of the Problem: Establishing the Extent of the Efficacy of the Intelligent Trust Evaluation Capability 4
      1.3.3 Delineation of the Problem: The Intelligent Evaluation of Trustworthiness 4
         1.3.3.1 Decomposition of the Problem 4
         1.3.3.2 A Common Question of Trust 5
         1.3.3.3 The Problem of Agent-Oriented Trust Evaluation 5
         1.3.3.4 The Problem of Evaluating Agent Hosts 6
      1.3.4 Summary 6
   1.4 Outline and Structure of Dissertation 6
   1.5 Conclusion 9

2. Introduction 11
   2.1 Introduction 11
   2.2 Agents 11
   2.3 Rational Agents 12
   2.4 Mobile Agents 13
      2.4.1 Migrating Autonomous Agents 13
      2.4.2 Mobile Agents in Context 13
2.4.3 Reasons for Mobile Agents 14
2.4.4 Degrees of Mobility 16
2.4.5 Summary 18

2.5 Mobile Agent Systems 18
2.5.1 Requirements for Mobile Agent Systems 18
2.5.2 Common Execution Language 19
2.5.3 Capturing Agent State 19
2.5.4 Inter-Host Communication 19
2.5.5 Security Requirements 19
2.5.6 Summary 20

2.6 Multi Agent Environments 21

2.7 Conclusion 22

3. Trust Theory 23

3.1 Introduction 23
3.2 What is Trust? 23
3.3 The Semantics of Trust 24
3.3.1 What it means to Trust 24
3.3.2 Accepting Risk 24
3.3.3 Relinquishing Control 25
3.3.4 Summary 25

3.4 The Beliefs of Trust 25
3.4.1 A Decomposition of Trust 25
3.4.2 Beliefs of Internal Trust 27
3.4.3 Beliefs of External Trust 29
3.4.4 Importance of Decomposition 29
3.4.5 Summary 30

3.5 Sources of Trust 30
3.5.1 Sources of Belief 30
3.5.2 Direct Experience 30
3.5.3 Reputation 30
3.5.4 Categorisation 31
3.5.5 Reasoning 31
4. Trust in Multi-Agent Systems: The State of the Art  
4.1 Introduction  
4.2 The Fuzzy Cognitive Map Approach  
4.2.1 Overview  
4.2.2 What is an FCM?  
4.2.3 Why Use FCM?  
4.2.4 Using FCMs for Trust Evaluation  
4.3 The Information Trust Model  
4.3.1 The Model in Context  
4.3.2 How Trust is modelled?  
4.3.3 Using an Information-Based Trust Model  
4.4 TRAVOS  
4.4.1 Overview  
4.4.2 TRAVOS: How Trust and Confidence are modelled  
4.4.3 TRAVOS: How Reputation is modelled  
4.4.4 Handling Inaccurate and Misleading  
Reputation Information  
4.5 REGRET  
4.5.1 Overview  
4.5.2 REGRET Components  
4.6 FIRE  
4.6.1 Overview  
4.6.2 FIRE Components  
4.7 Critical Evaluation  
4.7.1 Criteria for Evaluation  
4.7.2 Functional Criteria  
4.7.3 Non-Functional Criteria  
4.7.4 Evaluating the FCM Model  
4.7.5 Evaluating the Information Trust Model  
4.7.6 Evaluating TRAVOS  
4.7.7 Evaluating REGRET
4.7.8 Evaluating FIRE 81
4.7.9 Comparison of Evaluations 83
4.8 Conclusion 83

5. Trust and Reputation: The State of the Art 86
   5.1 Introduction 86
   5.2 Ontologies for Trust and Reputation 86
      5.2.1 Introduction 86
      5.2.2 A Functional Ontology of Reputation 87
      5.2.3 The Relevance of the Functional Ontology of Reputation 90
      5.2.4 Improvement and Future Work 91
      5.2.5 Summary 92
   5.3 Social Network Analysis & Relationship Analysis 92
      5.3.1 Introduction 92
      5.3.2 Social Network Analysis 93
      5.3.3 The Relevance of Social Network Analysis 93
      5.3.4 Summary 94
      5.3.5 Relationship Analysis 94
      5.3.6 The Relevance of Relationship Analysis 95
      5.3.7 Summary 96
   5.4 Incentive Compatible Reputation Mechanisms 96
      5.4.1 Introduction 96
      5.4.2 An Incentive Compatible Reputation Mechanism 97
      5.4.3 The Relevance of an Incentive Compatible Reputation
            Mechanism 98
      5.4.4 Summary 99
   5.5 Conclusion 99

6. Agent Architecture 101
   6.1 Introduction 101
   6.2 Agent Architecture 101
      6.2.1 An Overview 101
      6.2.2 BDI-Theoretic Architecture 103
6.2.3 Summary 104

6.3 Procedural Reasoning System Architectural Model 104
   6.3.1 An Overview 104
   6.3.2 Analysis 105
   6.3.3 Summary 106

6.4 JAM Architectural Model 106
   6.4.1 An Overview 106
   6.4.2 Analysis 107
   6.4.3 Summary 108

6.5 Nuin Agent Architecture 109
   6.5.1 The Nuin Agent Architecture in Context 109
   6.5.2 An Overview 109
   6.5.3 Analysis 111
   6.5.4 Summary 112

6.6 Modular BDI Theoretic Architecture 113
   6.6.1 Novak’s Modular BDI Agent Programming Architecture in Context 113
   6.6.2 An Overview 113
   6.6.3 Analysis 114
   6.6.4 Summary 115

6.7 Conclusion 115

7. Mobile Agent Systems: A Survey 117
   7.1 Introduction 117
   7.2 Mobile Agent Systems 118
      7.2.1 Mole 118
      7.2.2 D’Agents 119
      7.2.3 Aglets 120
      7.2.4 Concordia 121
      7.2.5 NOMADS 122
      7.2.6 Ajanta 123
      7.2.7 Summary 125
   7.3 Criteria for Quality Mobile Agent Systems 125
      7.3.1 The Relevance of Criteria 125
9.2.8 Controller 146
9.2.9 Trust Evaluation Components 147
9.2.10 Performance Evaluator 147
9.2.11 Learning Element 148
9.3 Conclusion 148

10. Desires, Intentions and Plans 150
10.1 Introduction 150
10.2 Desires and Intentions 150
10.2.1 A Primary High-Level Goal 150
10.2.2 Goal Decomposition 150
10.2.3 Analysing Goals 151
10.2.4 Summary 152
10.3 Plans to Minimise Betrayal 152
10.3.1 Overview 152
10.3.2 Plan Structure 153
10.3.3 Plan Selection 153
10.3.4 Trust Evaluation Meta-Plan 153
10.3.5 Establishing Direct Experience 156
10.3.6 Eliciting Reputation Information 158
10.3.7 Discovering Relationships 161
10.3.8 Detecting Betrayal 163
10.3.9 Evaluating Trustworthiness 166
10.3.10 Summary 167
10.4 Conclusion 167

11. Embedding a Specialised Trust Evaluation Capability 169
11.1 Introduction 169
11.2 A Component for Trust Evaluation 169
11.3 A Component for Attribution 172
11.4 A Bayesian Attribution Component 176
11.5 A Relationship Analysis Component 178
11.6 A Reputation Processing Component 181
11.7 Conclusion 181

12. Design and Implementation 183

12.1 Introduction 183
12.2 Approach 183
12.3 Specialised Trust Evaluator Agent Design 184

12.3.1 Introduction 184
12.3.2 Specialised Trust Evaluator Agent Class Model 186
12.3.3 Specialised Trust Evaluator Agent Controller Interaction Model 195

12.3.4 Summary 195

12.4 Deliberative Component Micro-Design 197

12.4.1 Introduction 197

12.4.2 Trust Evaluation Component Design 197
12.4.3 Reputation Processing Component 197

12.4.4 Relationship Analysis Component 199
12.4.5 Attribution Component 199

12.4.6 Bayesian Attribution Component 201
12.4.7 Summary 201

12.5 A Note on Implementation 203
12.6 A Working Prototype: REVALUATOR 206
12.7 Conclusion 220

Part III

13. Results 222

13.1 Introduction 222
13.2 Criteria for Analysis of REVALUATOR 222
13.3 A Qualitative Analysis 224
13.4 Metrics for Quantitative Analysis 228
13.5 Quantitative Analysis 232
13.6 Conclusion 244
14. Conclusion and Future Work 246
14.1 Introduction 246
14.2 Contribution 246
14.3 The Holistic Contribution of the Solution 249
14.4 Future Work 250
14.5 Conclusion 251

Appendix A 252
REVALUATOR Static Structure Diagrams 252

Appendix B 263
Ontologies 263
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 2.1</td>
<td>A Conceptual View of a Basic Agent</td>
<td>12</td>
</tr>
<tr>
<td>Figure 3.1</td>
<td>The Acceptance of Risk</td>
<td>24</td>
</tr>
<tr>
<td>Figure 3.2</td>
<td>Types of Trust, Reliance and Delegation</td>
<td>26</td>
</tr>
<tr>
<td>Figure 3.3</td>
<td>Characterisation of an Outcome</td>
<td>36</td>
</tr>
<tr>
<td>Figure 3.4</td>
<td>Attribution (adapted from [FAL04a])</td>
<td>37</td>
</tr>
<tr>
<td>Figure 3.5</td>
<td>Trust Atmosphere (adapted from [T3G])</td>
<td>42</td>
</tr>
<tr>
<td>Figure 3.6</td>
<td>Control aimed at Increasing the Global Trust</td>
<td>46</td>
</tr>
<tr>
<td>Figure 3.7</td>
<td>Beliefs of Contractual Trust</td>
<td>50</td>
</tr>
<tr>
<td>Figure 4.1</td>
<td>An Example FCM (adapted from [CAS03])</td>
<td>54</td>
</tr>
<tr>
<td>Figure 4.2</td>
<td>A Conceptual FCM Showing Beliefs and Belief Sources (adapted from [CAS03])</td>
<td>56</td>
</tr>
<tr>
<td>Figure 4.3</td>
<td>The Information Trust Model (adapted from [SIE05])</td>
<td>59</td>
</tr>
<tr>
<td>Figure 4.4</td>
<td>An Example Beta Probability Density Function</td>
<td>62</td>
</tr>
<tr>
<td>Figure 5.1</td>
<td>Types of ‘Reputative Knowledge’ (adapted from [CAS05])</td>
<td>88</td>
</tr>
<tr>
<td>Figure 5.2</td>
<td>An Architectural Model Involving the Functional Ontology of Reputation</td>
<td>90</td>
</tr>
<tr>
<td>Figure 6.1</td>
<td>The PRS Architectural Model (adapted from [BUS98])</td>
<td>105</td>
</tr>
<tr>
<td>Figure 6.2</td>
<td>The JAM Architectural Model (adapted from [HUB99])</td>
<td>107</td>
</tr>
<tr>
<td>Figure 6.3</td>
<td>The Nuin Agent Architectural Model</td>
<td>110</td>
</tr>
<tr>
<td>Figure 6.4</td>
<td>A Modular BDI-Theoretic Architecture (adapted from [DIC03])</td>
<td>113</td>
</tr>
<tr>
<td>Figure 8.1</td>
<td>Model for a Generic Trust Enhanced Mobile Agent System</td>
<td>136</td>
</tr>
<tr>
<td>Figure 9.1</td>
<td>Specialised Trust Evaluator Agent Architectural Model</td>
<td>142</td>
</tr>
<tr>
<td>Figure 10.1</td>
<td>Trust Evaluation Meta-Plan</td>
<td>154</td>
</tr>
<tr>
<td>Figure 10.2</td>
<td>A Plan to Establish Direct Experience</td>
<td>159</td>
</tr>
<tr>
<td>Figure 10.3</td>
<td>A Plan for Eliciting Reputation Information</td>
<td>161</td>
</tr>
<tr>
<td>Figure 10.4</td>
<td>A Plan for Relationship Analysis</td>
<td>162</td>
</tr>
<tr>
<td>Figure 10.5</td>
<td>Detecting Betrayal</td>
<td>165</td>
</tr>
<tr>
<td>Figure 10.6</td>
<td>Evaluating Trustworthiness</td>
<td>166</td>
</tr>
<tr>
<td>Figure 11.1</td>
<td>A Trust Evaluation Component</td>
<td>170</td>
</tr>
</tbody>
</table>
Figure 12.23  The Client’s Response to the
Specialised Trust Evaluator Agent’s Request 217

Figure 12.24  The Specialised Trust Evaluator Agent’s Response
to an Evaluation Request (A) 218

Figure 12.25  The Specialised Trust Evaluator Agent’s Response
to an Evaluation Request (B) 219

List of Tables

Table 4.1  A Tabular Comparison of Trust Models 84
Table 7.1  A Summary of Mobile Agent Systems 126
Table 7.2  A Qualitative Comparison of Mobile Agent Systems 130
Table 13.1  Target Agents Characteristics, Thresholds,
and Effective Techniques 229

List of Charts

Chart 13.1  Experiment 1 – Agent 1 233
Chart 13.2  Average Results for Experiment 1 234
Chart 13.3  Experiment 2 – Agent 2 235
Chart 13.4  Average Results for Experiment 2 235
Chart 13.5  Experiment 3 – Agent 3 236
Chart 13.6  Average Results for Experiment 3 237
Chart 13.7  Experiment 4 – Agent 4 238
Chart 13.8  Average Results for Experiment 4 238
Chart 13.9  Experiment 5 – Agent 5 239
Chart 13.10 Average Results for Experiment 5 240
Chart 13.11  Comparison of the Averages for Experiments 4 and 5 241
Chart 13.12  Experiment 6 – Agent 6 241
Chart 13.13  Average Results for Experiment 6 242
Chart 13.14  A Summary of Specialised Trust Evaluator
Agent Performance 243
# List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACL</td>
<td>Agent Communication Language</td>
</tr>
<tr>
<td>ATP</td>
<td>Agent Transfer Protocol</td>
</tr>
<tr>
<td>BDI</td>
<td>Belief-Desire-Intention</td>
</tr>
<tr>
<td>CORBA</td>
<td>Common Object Request Broker Architecture</td>
</tr>
<tr>
<td>DCOM</td>
<td>Distributed Component Object Model</td>
</tr>
<tr>
<td>DES</td>
<td>Data Encryption Standard</td>
</tr>
<tr>
<td>FCM</td>
<td>Fuzzy Cognitive Map</td>
</tr>
<tr>
<td>FIPA</td>
<td>Foundation for Intelligent Physical Agents</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>IDEA</td>
<td>International Data Encryption Algorithm</td>
</tr>
<tr>
<td>JADE</td>
<td>Java Agent Development Framework</td>
</tr>
<tr>
<td>JRE</td>
<td>Java Runtime Environment</td>
</tr>
<tr>
<td>JVM</td>
<td>Java Virtual Machine</td>
</tr>
<tr>
<td>KQML</td>
<td>Knowledge Query Manipulation Language</td>
</tr>
<tr>
<td>MAS</td>
<td>Multi-Agent System</td>
</tr>
<tr>
<td>OWL</td>
<td>Ontology Web Language/Web Ontology Language</td>
</tr>
<tr>
<td>PGP</td>
<td>Pretty Good Privacy</td>
</tr>
<tr>
<td>PRS</td>
<td>Procedural Reasoning System</td>
</tr>
<tr>
<td>RDF</td>
<td>Resource Description Framework</td>
</tr>
<tr>
<td>RMI</td>
<td>Remote Method Invocation</td>
</tr>
<tr>
<td>RPC</td>
<td>Remote Procedure Call</td>
</tr>
<tr>
<td>SSLv3</td>
<td>Secure Sockets Layer version 3</td>
</tr>
<tr>
<td>TCP/IP</td>
<td>Transmission Control Protocol/Internet Protocol</td>
</tr>
<tr>
<td>TRAVOS</td>
<td>Trust and Reputation model for Agent-based Virtual Organisations</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modelling Language</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Mark-up Language</td>
</tr>
</tbody>
</table>
Chapter 1

Problem Statement

1.1 Introduction
The concept of an agent (an entity that acts to achieve a goal with a degree of autonomy) provides an abstraction for handling the complexity of larger software systems. As such agent technology is an emerging paradigm for engineering complex software systems [JEN01].

As a consequence, the concept of a multi-agent system (MAS) has received a lot of attention in the not to distant past. More recently however, there has been a good deal of interest in a specialisation of MASs – namely mobile agent systems.

As with any emerging paradigm there are inevitably numerous issues that need ‘ironing out’. Trust and security in this context are regarded as key issues in need of attention [HU01, BOR02]. This is particularly true given the potential application of agent technology, and mobile agent systems in particular, to e-commerce.

The purpose of chapter 1 is to elaborate on and clearly define some of the aforementioned key issues and challenges facing the agent community. It is these aforementioned key issues and challenges that will be addressed in the subsequent chapters of the dissertation. Chapter 1 is arranged as follows: Section 1.2 provides the contextual background to the research problem to be addressed in the dissertation. Section 1.3 presents and analyses the research problem at hand. Section 1.4 outlines the structure of the dissertation. Section 1.5 provides a brief summary.

1.2 The Contextual Background of the Problem
1.2.1 Introduction
Section 1.2 provides the contextual background to the research problem that is to be addressed in the subsequent chapters of the dissertation. The following three sub-sections (sections 1.2.2 - 1.2.4) introduce and outline the problem in broad terms. The following three sub-sections also motivate the need for the research.
1.2.2 Evaluating the Trustworthiness of Agents
In recent years the problem of trust evaluation in the context of multi-agent environments has emerged. In many multi-agent environments numerous rational agents co-exist in collaboration and competition. Each agent typically has their own goals and performance measures [RUS03]. Furthermore, each agent has its own finite ‘sphere of influence’ [JEN01]. Given that the set of actions that define the agent’s so-called ‘sphere of influence’ must be finite, rational agents will almost certainly interact with each other - perhaps to achieve some global objective if not their own goals. Thus social relationships are likely to exist between the agents. A social relationship may require that one agent trusts another. The question is raised as to how one agent can trust another agent especially in competitive multi-agent environments where the performance measures and goals of different rational agents may conflict. Therefore, given the rationality of agents with distinct goals and performance measures it stands to reason that any given agent cannot be blindly trusted by default. It is clear at the outset that before a rational agent X trusts another agent Y, it is a requirement of agent X’s rationality for it to consider the trustworthiness of agent Y. The question of trust is extended when the agents in a system are able to move between unknown hosts.

1.2.3 Evaluating the Trustworthiness of Agent Hosts
When agents have the capacity to move between hosts an additional complication is introduced. The migrating agent is at the mercy of the potentially malicious host at which it will arrive. Indeed perhaps the greatest impediment to the application and use of the mobile agent paradigm is the concern over mobile agent security [BOR02]. The mobile agent security issue is a bilateral one. Agent hosts are under threat from malicious agents whilst mobile agents are under threat from malicious hosts, malicious users and their agents [BIE02, KOT02]. The latter of the aforementioned problems is proving particularly difficult to solve, to the point that there are concerns that a complete solution may not be possible [CLA03, VIG04, WAN02]. The difficulties associated with mobile agent protection arise as a result of the partial ineffectiveness of traditional security mechanisms when applied to the mobile agent paradigm [FAR96]. It is expected that by considering the trustworthiness of agent

---

1 An example of agent interaction (where inter-agent trust is a prerequisite) to achieve a global objective can be found in the literature [BER01].
hosts, both new and traditional security mechanisms can be augmented resulting in more secure mobile agent systems. Nonetheless, the extent of the effectiveness of a trust-based approach aimed at augmenting existing mechanisms remains an open question.

1.2.4 The Need for an Intelligent Approach to Trust
As already indicated in the above sections an intelligent approach to trust is necessary. For an intelligent agent to blindly rely upon (or trust in) other agents and/or agent hosts is generally not an intelligent way to act. There are few cases where the action to trust other agents without reason, could be seen as rational; one such case may arise if the agent had no choice (or almost no choice) and risked serious loss through failure. Furthermore, it is generally in an agent’s interest to engage in interactions with other agents; cooperating with other agents could be profitable for an agent. However, this needs to be done carefully and hence intelligently. Clearly the need exists for agents to be able to reason about the trustworthiness of other agents (both software agents and human agents) as well as agent hosts.

1.2.5 Summary
Section 1.2 contributes by highlighting the requirement for a rational agent to rationally consider the trustworthiness of the entities (other agents and agent hosts) in its environment prior to any decision to rely on the entity in question.

Through careful analysis, the problem to be addressed in the dissertation can be specified more precisely. Moreover, the same research problem can be decomposed into a set of related research questions. The precise specification and delineation of the problem is the content of section 1.3 to follow.

1.3 The Problem Statement
1.3.1 A Concise Problem Definition
Following the introduction and outline of the problem in the previous three sections, it is now possible to concisely define the problem:
It has been established that rational agents have a need to give rational consideration to the trustworthiness of various entities (be it other agents or agent hosts). How does a rational agent intelligently determine the trustworthiness of another entity (where an entity refers to either an agent or an agent host) and to what extent is that trust evaluation capability effective?

The concise problem definition posed above highlights two related issues. The former of the related issues involves the determination of trustworthiness and is addressed in sub-section 1.3.3. The latter of the aforementioned issues involves determining the extent to which the trust evaluation capability is effective. The latter issue is addressed in section 1.3.2.

1.3.2 Delineation of the Problem: Establishing the Extent of the Efficacy of the Intelligent Trust Evaluation Capability

Determining the extent of the efficacy of the trust evaluation capability implies that the purpose of the research presented herein is to develop an intelligent trust evaluation capability that allows for the evaluation of various entities (agents and their hosts). Furthermore, it is clear that the research presented in the dissertation must also provide both qualitative and quantitative metrics for measuring the extent of the effectiveness of the intelligent trust evaluation capability. Given the interest in determining the effectiveness of the intelligent trust evaluation capability it seems reasonable to suggest the following hypothesis that will be tested in the dissertation:

Given that intelligent agents are sometimes betrayed by other intelligent agents, limits to the effectiveness of agent-oriented trust-based approaches do exist. Nonetheless agent-oriented trust-based approaches are of some value in limiting incidents of agent betrayal.

1.3.3 Delineation of the Problem: The Intelligent Evaluation of Trustworthiness

1.3.3.1 Decomposition of the Problem

The former of the issues highlighted by the concise problem definition posed previously in sub-section 1.3.1 must be decomposed into a series of research questions.
The main problem at hand is the intelligent evaluation of the trustworthiness of another entity. There are two issues here:

- Firstly, what does the intelligent evaluation of an agent’s trustworthiness involve?
- Secondly, what does the intelligent evaluation of an agent host’s trustworthiness involve?

The paragraphs that follow decompose the above issues into a set of research questions to be addressed in the dissertation.

1.3.3.2 A Common Question of Trust
The common question posed above is how to reach an intelligent evaluation of trustworthiness. To address the question of how to evaluate trustworthiness, it is necessary to investigate the concept of trust. In particular, what does it mean to trust and what does trust and deciding to trust involve? The research presented herein must therefore thoroughly investigate the semantics of trust.

1.3.3.3 The Problem of Agent-Oriented Trust Evaluation
With respect to the particular question of evaluating another agent’s trustworthiness, it is both necessary and prudent to consider what methods/approaches/models currently exist for the purpose of evaluating the trustworthiness of agents. The research presented herein must therefore investigate methods/approaches/models for agent-oriented trust evaluation and provide insight into their functionality. Moreover, the research must ask and answer what makes a method/approach/model fit for the purpose of agent-oriented trust evaluation. By answering the question of suitability, it becomes possible to analyse and compare methods/approaches/models aimed at agent-oriented trust evaluation.

With a view to providing a solution to the question of evaluating another agent’s trustworthiness, the research presented in the dissertation must identify architectural abstractions suitable for agents capable of trust. In order to identify suitable architectural abstractions it is necessary to evaluate existing agent architectural styles. It is a challenge of the research presented in the dissertation to employ the identified architectural abstractions in proposing an architectural model that facilitates and
supports trust evaluation. In keeping with elegant software engineering practice, it is also a challenge of the research presented herein to abstract and distil the most fundamental principles of agent-oriented trust evaluation and package them as a reusable component (or set of components).

1.3.3.4 The Problem of Evaluating Agent Hosts
It is expected that by addressing the question of agent-oriented trust evaluation as posed in the previous paragraph, a solution to the particular problem of how to evaluate an agent host’s trustworthiness will become apparent. It is expected that the fundamental principles of trust and the similarity of the two problems will allow for a shared solution by means of extrapolation and generalisation. Nonetheless, aspects of agent mobility and mobile agent systems cannot be ignored if a feasible solution is to be produced. The research presented herein must therefore also investigate mobile agents, their mobility and mobile agent systems in general. The research presented herein must also try to establish if certain commonalities exist amongst the mobile agent systems to be surveyed. Assuming the existence of commonality amongst mobile agent systems it is expected that an architectural model aimed at augmenting existing ‘hard’ mechanisms for mobile agent protection with a ‘soft’ trust-based approach can be specified. The specification of the aforementioned architectural model is a desired contributing output of the research presented in the dissertation.

1.3.4 Summary
Section 1.3 contributes by clearly delineating the problem that the research presented in the dissertation attempts to address. A top-down approach has been adopted in decomposing the concise problem statement of section 1.3.1 into a set of related research questions.

The section that follows outlines the structure of the dissertation and is intended to give the reader an overview of the work to follow.

1.4 Outline and Structure of Dissertation
The previous sections have posed the questions that the research presented in this dissertation aims to answer. The dissertation can be seen as comprising three
constituent parts. Each part consists of several chapters. The dissertation is arranged as follows:

**Part I**

Broadly speaking, Part I deals with the theory and concepts abstracted from the reviewed literature. The concepts and contributions of the reviewed literature are summarised and analysed. The chapters comprising Part I typically contribute by answering the posed research questions and/or by laying a foundation on which to build on in Part II. The following chapters constitute Part I:

**Chapter 2** serves as an introduction to the important concepts of agency. Some of these concepts have already been used in the formulation and delineation of the problem. The chapter introduces the concept of a rational agent. It also explains what is meant by a mobile agent. Chapter 2 also deals with multi-agent systems in general and mobile agent systems in particular. The content of chapter 2 is necessary for the purpose of clarification. It also contributes by providing a necessary foundation on which to build in the latter chapters.

**Chapter 3** presents an important theory of trust. Chapter 3 sheds some light on what it means to trust and answers the question of how one agent may trust another. In so doing chapter 3 introduces important trust related concepts and theory that are critical to solving the research problem and answering the questions posed. Not only does chapter 3 contribute by directly addressing some of the research questions posed, but it is also a foundational chapter. Many of the concepts presented in chapter 3 are endemic throughout the dissertation.

**Chapters 4 and 5** form part of an investigation into the state of the art. In particular, chapter 4 focuses on existing trust models, whilst chapter 5 focuses on supplementary contributions. The work provided in chapter 4 provides insight into the functionality of existing trust models. Chapter 4 also suggests criteria for analysing the suitability of a method or model aimed at trust evaluation.

**Chapter 6** pays attention to the topic of agent architecture. Various architectural models are investigated and analysed with a view to elicit the most relevant
architectural abstractions for the purpose of producing agents with the capability to evaluate trust.

Chapter 7 investigates mobile agent systems in accordance with the requirements of the research as set out in the previous section. Perhaps most notably, chapter 7 analyses and compares the surveyed mobile agent systems. The contribution lies in the establishment of some commonality amongst mobile agent systems.

Part II
The work presented in Part II details the application of the theory and concepts presented in Part I in providing a solution to the research problem. The chapters comprising Part II present a model (and the implementation thereof) aimed at solving the research problem.

Chapter 8 presents a high-level architectural model for a trust enhanced mobile agent system and in so doing provides the context for the work presented in chapter 9. The contribution of the chapter lies in the specification of a scheme that allows for agents in a mobile agent system to benefit from the protection offered by a trust-based approach.

Chapter 9 presents a specialised architectural model for a specialised trust evaluator agent based on architectural abstractions previously identified as relevant. The trust evaluating agent is a service providing entity that is in itself an abstraction used in the architectural model presented in chapter 8.

Chapter 10 builds on chapter 9 by providing relevant details relating to the evaluation of trustworthiness. Chapter 10 presents the desires, intentions and plans of the specialised trust evaluator agent. Chapter 10 is a necessary ‘stepping stone’ to chapter 11 as it provides insight into the specialised trust evaluator agent’s plans. The specialised trust evaluator agent’s plans represent the procedures involved in trust evaluation. The trust evaluation plans bootstrap functionality provided by the components that are the topic of chapter 11.
Chapter 11 concludes the model chapters by presenting the high-level design of the components that embed a specialised trust evaluation capability. Chapter 11 goes along way to meeting the goals of the research as set out in the previous section, as it presents an effort to abstract and distil the important concepts of trust evaluation and package them as a set of related components.

Chapter 12 presents the design and implementation details of a prototype implementation of the model. Chapter 12 presents a prototype implementation that serves as proof of concept. Chapter 12 also provides the reader with some insight into the working of the prototype.

Part III
The chapters comprising Part III present and critically analyse the results produced through the efforts outlined in Part II. Part III also includes the conclusion of the research and thus highlights both the specific and holistic contributions of the research effort.

Chapter 13 thoroughly analyses the solution (using the implemented prototype where possible). Chapter 13 presents quantitative results produced using the implemented solution. A major contribution of chapter 13 lies in the quantification of the extent of the effectiveness of the implemented trust evaluation capability.

Chapter 14 concludes the dissertation by summarising the contribution of the work and highlighting areas for future research.

1.5 Conclusion
Chapter 1 introduces several problems that hinder the use of agent technology in some cases. The challenge of evaluating trustworthiness of various entities including both agents and agent hosts needs to be overcome. Therefore, to summarise: the purpose of the research presented in the dissertation (as specified in section 1.3) is to investigate, design, develop and evaluate an intelligent trust evaluation capability for the purposes of evaluating agents and agent hosts. In so doing, existing ‘hard security’ mechanisms
aimed at mobile agent protection will be augmented with a ‘soft’ trust-based approach.

Chapter 1 contributes by firstly providing the necessary contextual background to the problem (section 1.2). The major contribution of the chapter lies in the delineation of the problem to be addressed in the dissertation. A concise problem statement is provided as well as a set of research questions stemming from the analytical decomposition of the problem statement (section 1.3). Chapter 1 also provides the reader with an overview of the dissertation, particularly highlighting its structure (section 1.4).
Chapter 2
Introduction

2.1 Introduction
Chapter 1 discussed some challenges associated with agent technology. However, no mention was made as to what an agent actually is. Nor was any attention paid to the concepts of mobile agents and mobile agent systems.

Chapter 2 serves as a brief introduction to agent technology, mobile agent systems and multi-agent environments. Basic key concepts are presented that are very important in the context of the dissertation. The remaining chapters assume knowledge of these concepts. This chapter begins with a brief introduction to agents. Section 2.3 elaborates on the introduction by discussing rational agents. In section 2.4 the focus shifts toward mobile agency. Sections 2.5 and 2.6 discuss mobile agent systems and multi-agent environments respectively.

2.2 Agents
In chapter 1 reference to agents, and agent technology in general was made. An agent is an entity that perceives its environment through its sensors and acts in that environment through its actuators [RUS03]. Jennings describes software agents as ‘clearly identifiable problem solving entities’ [JEN01]. Furthermore, he goes on to say that agents are embedded in their environments and should be capable of autonomous, proactive and reactive behaviour in that environment. Figure 2.1 provides a conceptual view of an agent.

As Jennings asserts – an agent is situated in its environment [JEN01]. For the agent to solve some problem in that environment it must be able to perceive that environment (or part of it) and effect some change on it. Software agents may perceive messages, keystrokes, file contents, etc. Similarly, such agents may act on their environment through sending messages, writing to files, displaying information on a console, and so forth.
For software agents to be autonomous, proactive, and reactive suggests that they should be ‘intelligent’ [BOR02, JEN01]. Agents may be ‘intelligent’ because ‘intelligence’ is embedded within these agents. That is to say that through software engineering efforts these agents are given the ability to act intelligently. So what exactly does it mean for an agent to act intelligently? This depends on how one defines intelligence and artificial intelligence. In this document acting intelligently is intended to mean acting rationally. This is consistent with the view of Russell and Norvig [RUS03]. The rational agent approach is discussed next.

Figure 2.1 A Conceptual View of a Basic Agent

2.3 Rational Agents
Russell and Norvig present a rational agent approach in their book [RUS03]. They define a rational agent as an agent that for any percept sequence selects an action that is expected to maximise its performance measure, given what can be deduced from the percept sequence and what it knows. This means that rational agents should always do the right thing given their knowledge.
Successful rational agents require careful design. They should also have the capacity to learn and should be autonomous. However, not all agents are endowed with such abilities. Some agent programs may function using only condition-action rules (simple reflex agents). Russell and Norvig outline several basic models for agents in their book. Included in this are models for simple reflex agents, model-based reflex agents, goal-based agents and utility-based agents [RUS03].

The above paragraph introduces the concept of a rational agent. Rational agents act intelligently and thus it is hoped will exhibit a high degree of autonomy, as well as a capacity for both proactive and reactive behaviour. The next section introduces a specialisation of agents – namely mobile agents.

2.4 Mobile Agents

2.4.1 Migrating Autonomous Agents

In section 2.3 on rational agents, it was suggested that (ideally speaking) rational agents ought to be autonomous. Indeed, some agents that exhibit such autonomy may migrate to other hosts in a network as and when they choose. These agents are commonly referred to as mobile agents [BOR02]. The following paragraphs offer an in-depth introduction to mobile agents.

2.4.2 Mobile Agents in Context

This paragraph puts mobile agent technology into context. Mobile agency is amongst a group of competing technologies that deliver the same outcomes [REI98]. These technologies can be classified into the following categories:

- Remote procedure call (RPC) paradigm.
- Traditional client-server paradigm.
- Message passing paradigm.

These three paradigms (along with mobile agency) all share a common characteristic; typically an instruction is executed that results in execution of code on a remote host. However, the lines between these categories can be a bit blurred; RPC technologies can be used to implement a client-server system and RPC can be described as message passing.
The message passing paradigm discussed here actually refers to a protocol or a language - the commands or sentences of which is passed between nodes in the system. An example of such a technology is the Knowledge Query Manipulation Language (KQML) [FIN97].

An extension of the RPC paradigm (distributed object architectures) can also be used to implement mobile agent systems. Java’s Remote Method Invocation (RMI) is one such example [BAU98]. Other distributed object architectures include the Common Object Request Broker Architecture (CORBA), Distributed Component Object Model (DCOM) and more recently .NET Remoting [COM04]. It is foreseeable that any of the above mentioned distributed object architectures could be used in the implementation of mobile agent systems.

2.4.3 Reasons for using Mobile Agents

The above paragraph discussed several competing technologies. This paragraph motivates the use of mobile agents. Mobile agents provide several benefits when used to create distributed systems. Lange and Oshima suggest seven benefits of using mobile agents to this end [LAN99]. These benefits are briefly discussed:

The use of mobile agents may reduce network traffic [LAN99, BOR02]. This will be the case if for a given transaction the mobile agent’s code and state is less (in bytes) than the volume of data that would have to be transmitted if another competing technology was used. Typically, if large volumes of data on a remote host have to be transmitted then it may be more efficient to move the computation to the data - meaning that the migration of the mobile agent to the remote host where the data is stored may result in reduced network utilisation.

Mobile agents can support seamless system integration [LAN99]. This is because mobile agents are typically only dependant on their execution environments. Mobile agents may work at the application level and are transport layer independent. This is important since network computing is fundamentally heterogeneous.

Mobile agents also have the ability to adapt to their environment. So, in a networking context, if there is a problem with one host the agent may simply move to another and
service availability is maintained. This also introduces a level of robustness which makes mobile agent systems relatively fault tolerant [LAN99].

Large, distributed, real-time systems can benefit from mobile agent technology; in such a system network latency may be problematic. Mobile agents can be employed to solve such a problem because they can migrate from a central controller to a remote host to act locally [LAN99].

Mobile agents can also be used to solve another problem. As communication needs change, the protocols used sometimes become a legacy problem. Mobile agents can encapsulate protocols [LAN99]. Thus, they can move to remote hosts, establish communication channels based on existing protocols and then the new protocols that are encapsulated may be used.

Another significant advantage of mobile agent technology is that mobile agents may execute asynchronously and autonomously [LAN99]. This means that a task can be embedded within an agent and the agent can then migrate between hosts in the network while executing the task. During this time, the owner or user of the agent may disconnect from the network.

In addition to the above benefits of mobile agent technology (stemming from the fundamental nature of mobility) Lange and Oshima also identify several application domains that could benefit from the application of mobile agents. The application domains mentioned include E-commerce, personal assistance, secure brokering, distributed information retrieval, telecommunication network services, groupware, monitoring and notification, information dissemination and parallel processing. These are discussed in [LAN99].

Section 2.4.3 briefly outlines the benefits of mobile agent technology. A list of application domains that could benefit from mobile agency is also provided. The following section focuses on the nature of mobility.
2.4.4 Degrees of Mobility

It has been mentioned that mobile agents may migrate between hosts. So what does this actually mean? This section discusses the more general principle of mobile code. Different types of agent mobility are also discussed.

Code mobility refers to program code that moves from one host to another. Code mobility is not that uncommon. In client-server computing for example, a Java applet or ActiveX control may be downloaded from a Web server to a client computer. The code comprising the control or the applet then executes on the client machine. There are two common methods of code mobility: remote execution and code on demand [BAU98, VIG04].

In the case of remote execution, program code is transferred before its execution to a remote host. The code executes on the host until it terminates. In the context of mobile agency, one agent already running on a host could start the execution of other agents using remote execution. This means the agent transmits the agent program (and possibly some additional parameters) to a remote host where the agent program will begin execution [BAU98].

Code on demand differs from remote execution in that the remote host initiates the transfer of program code. In the context of mobile agency, the remote host may request the transfer of the agent program. Once transferred, the agent program will begin its execution [BAU98, VIG04].

So far, only code mobility has been discussed. The focus now shifts to agent mobility. Agent mobility differs from code mobility in that as far as code mobility is concerned, only program code is transferred. In the case of agent mobility, the agent program as well as the agent state is transferred. Thus code mobility is more general than agent mobility. Agent mobility is a type of code mobility but not vice versa. There are two common approaches to agent mobility: strong migration and weak migration [BAU98, KOT02, VIG04].

Strong migration is the highest degree of mobility. In this case an autonomous mobile agent will make a request to the underlying system to migrate to another host. The
underlying system will then transmit the agent program and the current agent state to the new host.

The agent state can be considered to comprise both the data state and the execution state of the agent. The data state consists of the values of global and instance variables. The execution state consists of the values of the program counter and the registers, as well as the data in the process stack. In the case of a single threaded process the process stack contains all method parameters, return addresses, and local variables. If the process is multi-threaded then each thread will have its own stack and its own set of registers [SIL04].

Thus in the case of a multi-threaded mobile agent and a mobile agent system supporting strong migration there is much data to be transferred. This can impair at least one benefit of using mobile agents as discussed in section 2.4.3. The use of mobile agents may not reduce the volume of data to be transmitted across the network. Other disadvantages of this approach as identified in [BAU98] include requiring a global model of agent state and transfer syntax. This requirement exists because of the heterogeneity in a distributed system and due to the fact that the agent has to be reassembled after transmission to the new host. One advantage of strong migration is that the programmer need not worry about transfer and restoration of the agent because this is handled by the underlying system [BAU98].

Due to the difficulties associated with strong migration, the principle of weak migration has emerged. Weak migration differs from strong migration in that only the data state (and not the execution state) is transferred. Thus there must be some mechanism to allow execution to continue from the appropriate point after transmission. To achieve this, the programmer must store any critical execution state information in global variables. Additionally the programmer must provide some sort of start method that allows the agent to determine where to continue its execution on the new host. Obviously this approach requires more effort from the programmer but can reduce the amount of data to be transmitted [BAU98].
Section 2.4.4 discussed two approaches to pure code mobility and two approaches to a more specialised form of code mobility – agent mobility. The following paragraph summarises the work presented here.

2.4.5 Summary
Section 2.4 introduced mobile agents as autonomous agents that migrate to other hosts in a network as and when they choose. The mobile agent paradigm was placed into context by briefly drawing attention to competing technologies that can be classified as falling into one of the following categories: RPC paradigm, traditional client-server paradigm, and the message passing paradigm. Reasons for using mobile agent technologies were also highlighted. Application domains that could benefit from mobile agency were listed. Finally, degrees of mobility and the differences between code mobility and agent mobility were discussed.

Section 2.4 is of primary relevance because it provides a foundation for the discussion and analysis of mobile agent systems in chapter 7. Moreover, a holistic understanding of mobile agency and how mobility is achieved may be useful given the ultimate goal of protecting mobile agents from malicious hosts – albeit via a trust-based approach.

2.5 Mobile Agent Systems
2.5.1 Requirements for Mobile Agent Systems
Section 2.4 discussed mobile agents. The focus of this section is systems that provide the infrastructure to make mobile agency possible.

A mobile agent system is a system that provides an environment for mobile agents to operate in. A mobile agent system consists of the underlying supporting software environment and the mobile agents in the environment. Mobile agent systems need to satisfy at least four functional requirements [REI98]:

- A common execution language must be provided.
- It must be possible to capture an agent’s state.
- Agent hosts must be able to communicate.
- Security to protect hosts and agent hosts.

Each of the above mentioned requirements is now discussed.
2.5.2 Common Execution Language

For processes (code in particular) to migrate between hosts in a distributed, heterogeneous environment a common execution language seems desirable. While it may be possible to transmit native machine code between hosts this would require some translation of the machine code if the two hosts have different system architectures.

Therefore, to ease implementation of mobile agent systems it is beneficial to use a common execution language that targets a common virtual machine. This is the reason the Java programming language and the Java Runtime Environment (JRE) is a popular choice in developing mobile agent systems. In such a system the Java byte code is transmitted between hosts, each running an instance of the Java Virtual Machine (JVM). The Java interpreter (part of the JVM) translates the byte code to native machine code for each host.

2.5.3 Capturing Agent State

The above paragraph discusses the need to transmit code between hosts. However, there is more to a process than code. The mobile agent system must have a way to capture the state of the agent (at least the data state of the agent process). In the case of a mobile agent system supporting strong migration the execution state as well as the data state need to be captured. The state then needs to be transmitted to a remote host.

2.5.4 Inter-Host Communication

In order to transmit an agent (both the agent code and the agent state) between hosts, the hosts need to be able to communicate. The communication may be facilitated using existing protocols such as the Transmission Control Protocol/Internet Protocol (TCP/IP) or an application level protocol such as the Hypertext Transfer Protocol (HTTP).

2.5.5 Security Requirements

The usual security concerns when transmitting data across a network also exist. The confidentiality and integrity of the agent code and state need to be protected both
during transmission and at the remote host. Reasons for agent protection include [WIL98, BIE02]:

- Agents may carry sensitive, private data that should be protected.
- Unauthorised tampering with agent code or agent state (including data state) could result in the agent not behaving as it should.

The above mentioned concerns require that the agent be protected. The above mentioned concerns are not the only concerns however. A classification of the threats faced by mobile agents from malicious hosts is provided in the literature [BIE02].

It is also necessary to protect the agent host. This is partly due to the last concern mentioned above; if the integrity of the agent’s code or state is compromised then the agent could act in a damaging way at the remote host. Furthermore, it is also possible that a malicious agent could be used to attack a remote host.

While the previously outlined security issues are perhaps the most challenging of security concerns in the mobile agency context, they are not however, the only security concerns. To ensure the proper functioning of a mobile agent system, agents and the agent hosts need to be available. It is also the case that authorisation, non-repudiation, identification and authentication concerns will need to be addressed in any real-world application of mobile agents, particularly in e-commerce domains.

Sub-section 2.5.5 considers security requirements that mobile agent systems should attempt to address. A summary of this section now follows.

**2.5.6 Summary**

Section 2.5 introduces mobile agent systems by discussing four important requirements that a mobile agent system should satisfy. The need for a common execution language, capturing agent state, inter-host communication, and various security requirements were discussed. The basic functionality that ought to be provided by a mobile agent system has been discussed.

The contribution of section 2.5 is again in the provision of a foundation for a thorough discussion and analysis of mobile agent systems in chapter 7. Furthermore, the requirements of mobile agent systems as presented here suggest certain areas of
commonality amongst mobile agent systems - assuming that mobile agent systems do in fact meet the specified requirements. Areas of commonality amongst mobile agent systems may prove of interest given the desirability for a generic trust-based solution that can be applied in a range of mobile agent systems.

2.6 Multi Agent Environments

The previous section discussed mobile agent systems. Any mobile agent system containing more than one agent may also be referred to as a multi-agent environment. Section 2.6 discusses multi-agent environments in general.

As mentioned earlier in chapter 2, an agent is used to solve a particular problem. In adopting an agent-oriented philosophy or approach to system development it is the typically the case that large complex problems can be decomposed into smaller more manageable problems [JEN01]. Therefore, the agent oriented paradigm more often than not involves the use of multiple agents in meeting the demands of large complex problems. The use of multiple agents to solve a problem results in a solution known as a multi-agent system. A multi-agent system is simply an environment where agents have to interact in order to solve a problem.

An agent may only be able to view a part of the environment. Similarly, the agent may only be able to effect change on a part of the environment. The limitations on perception and action correspond to so-called spheres of visibility and influence in the cited literature [JEN01]. Since different agents may solve different problems these spheres of visibility and influence may overlap. Thus agents may need to interact to coordinate their activities. Also, given the limitations of what an agent is able to see and do it is necessary for the agent to interact with other agents in order to achieve its goals [BOR02, JEN01]. Thus there clearly are social relationships among the agents in a multi-agent environment.

The nature of the social relationships can be the basis for classifying the multi-agent environment. Indeed, Russell et al distinguish between cooperative and competitive environments [RUS03]. In the same literature they further specify various criteria for classifying the agent’s environment.
This section serves as a brief introduction to multi-agent environments. This chapter is now concluded in the remaining section.

2.7 Conclusion
This chapter introduces some fundamental concepts of importance for later chapters. These concepts include agents, mobile agent systems and multi-agent environments. Section 2.2 introduces agents as entities that perceive and act. Rational agents are also discussed. Rational agents always do the right thing given their knowledge. An intelligent agent is a rational agent. Mobile agents are also discussed. Mobile agents exhibit sufficient autonomy so as to be able to migrate between hosts in a network. Mobile agents systems are presented as systems that provide the infrastructure to make mobile agency possible. Multi-agent environments are also discussed.

The importance of the work as summarised above lies in the provision of a solid foundation from which the problems and challenges laid out in chapter 1 can be examined. Furthermore, insight into current attempts at solutions can be investigated. The next chapter presents a theory of trust.
Chapter 3
Trust Theory

3.1 Introduction
Chapter 1 mentioned the importance of trust in mobile agent systems and multi-agent environments. Chapter 3 examines trust in detail. The theory presented in chapter 3 serves as a foundation upon which a trust component can be developed. The presented trust theory originates from the cognitive approach to trust as presented in the reviewed literature [CAS98, CAS00, CAS01a, CAS01b, CAS01c, CAS03, FAL04a, FAL04b, T3G]. Chapter 3 summarises (and in some instances augments) the main ideas in the above mentioned work.

3.2 What is Trust?
In defining trust it is important to identify what trust is not. Trust is not simply a number or a probability. Trust in an agent may ultimately be represented by a number or a probability but trust is far more complicated than such a representation would suggest [CAS98, CAS00, CAS01b, T3G].

Trust is a mental attitude [CAS98, T3G, McL06]. Trust in its truest sense is an agent’s belief that it will not be betrayed by another agent with respect to some task or goal. A definition of trust is discussed below.

‘Trust is a mental state, a complex attitude of an agent X towards another agent Y about the behaviour/action α relevant for the result (goal) g’ [CAS98]. The definition indicates that trust is a mental attitude. The definition implies the cognitive nature of trust. It is because of the cognitive nature of trust that only a cognitive agent is capable of trusting.

Section 3.2 provides a definition of trust. Subsequent sections build on the definition giving the reader a clear and concise picture of trust. The definition presented here is not very descriptive in that it does not convey precisely what is meant by trusting. The following section attempts to convey the semantics of trust.
3.3 The Semantics of Trust

3.3.1 What it means to Trust
Section 3.3 elaborates on the previous section by explaining what is meant by an agent X trusting an agent Y with respect to some task \( \alpha \). Paragraph 3.3.2 discusses the acceptance of risk in trusting an agent. Paragraph 3.3.3 discusses the relationship between trust and control.

3.3.2 Accepting Risk
For an agent X to trust an agent Y means that X must accept some risk [McL06, CAS98, CAS00]. Trust as a belief implies uncertainty. If there was no uncertainty then the belief would in fact be knowledge. Thus because of the implied uncertainty, risk does exist. The risk is the risk of being betrayed by Y in the event that Y fails to complete the delegated task \( \alpha \) [CAS98, CAS00, McL06]. Thus, X must accept the risk of failure – the risk that X may not be able to achieve its goal [CAS00]. X must also accept the risk of incurring some loss as a result of betrayal in the event that it occurs. The loss referred to is at the least the loss incurred as a consequence of wasted effort, but may also include additional loss as a consequence of failure (such as the loss incurred because of a penalty imposed in the event of failure) [CAS00]. Figure 3.1 illustrates the risk accepted by agent X.

![Figure 3.1 The Acceptance of Risk](image-url)
The agent X expends time and effort (resources) in evaluating the trustworthiness of agent Y. In the event that agent X is betrayed, the evaluation was a waste of time and resources and therefore the betrayal amounts to the wasted effort mentioned above.

3.3.3 Relinquishing Control
If X trusts Y it is implied that X will not attempt to complete $\alpha$ itself, nor will X attempt to delegate $\alpha$ to another agent [CAS98, CAS00]. The implication of agent X’s trust in agent Y is that agent X will relinquish control; if X truly trusts Y then X will not need to control Y with respect to task $\alpha$. If X trusts Y, X believes Y is capable and will do $\alpha$ [CAS00]. Therefore, where there is trust there is no room for control and where there is control there is no room for trust. While there is truth in the above conclusion it is however, an oversimplification of the relationship between trust and control [CAS00]. The relationship between trust and control is far more complicated than it would seem and is discussed in section 3.9.

3.3.4 Summary
Section 3.3 emphasises the meaning of trust. It is clear that for an agent to trust another, the agent must accept the risks stemming from the inherent uncertainty involved in trusting the other agent. One of the risks referred to is the risk of being betrayed. As a consequence of the risk of betrayal the agent further risks failure and loss. Furthermore, an agent must necessarily give up control of the task to be delegated if the agent indeed trusts the agent to which the task will be delegated. The relationship between trust and control is discussed later in chapter 3. Section 3.3 contributes to the dissertation by addressing the semantics of trust. The following section looks at the beliefs of trust.

3.4 The Beliefs of Trust
3.4.1 A Decomposition of Trust
Section 3.2 provides a generic, rather abstract definition of trust. Section 3.3 attempts to add some substance to the notion of trust by discussing the semantics of the concept. However, the notion of trust as presented up until this point is still quite vague. Section 3.4 strives to eliminate the vagueness by decomposing trust into its core constituents. Clarity is introduced through the presentation of a top-down
decomposition of the notion of trust. According to Castelfranchi and Falcone, trust can be considered as consisting of internal trust and external trust [CAS00, CAS01b]. The different types of trust, their constituent beliefs, along with the concepts of reliance and delegation are illustrated in figure 3.2 below, and discussed in the following paragraphs. The importance of the decomposition is highlighted in the penultimate paragraph.

Figure 3.2 Types of Trust, Reliance and Delegation

The general use of the term ‘trust’ refers to a broad notion of trust. The broad notion of trust or ‘global trust’ is the trust that an action or task will be executed successfully or that a goal will be achieved [CAS00, CAS01b, T3G]. ‘Global trust’ is essentially a fulfilment belief - the belief that a task $\alpha$ will be executed successfully [CAS01b, T3G]. The fulfilment belief is supported by, or based on two other forms of trust and the beliefs that comprise them. The one form of trust is the trust that an agent X has in an agent Y [CAS00, CAS01b, T3G]. The trust in an agent corresponds to the internal
trust. The other form of trust is the trust that agent X has in the environment. The trust in the environment corresponds to the external trust [CAS00, T3G]. The beliefs of both X’s trust in Y and X’s trust in the environment are the subject of the paragraphs that follow.

3.4.2 Beliefs of Internal Trust
X’s trust in Y can be seen as a combination of at least three primary beliefs. The three primary beliefs are supported by other, more specific beliefs. The first of the beliefs is a competence belief. For X to trust in Y with respect to some task \( \alpha \), X must believe that Y is competent with respect to \( \alpha \). Agent X’s competence belief implies that X believes that Y has the knowledge to complete \( \alpha \) successfully [CAS98, CAS00, CAS01b, FAL04a, McL06, T3G]. X may also believe that Y has the necessary experience and skill to complete \( \alpha \). X must also believe that Y is confident of its ability to complete \( \alpha \) successfully [CAS01b, T3G]. Agent X’s belief of agent Y’s confidence is known as the self-confidence belief but it only applies in the case that Y is a cognitive agent. The self-confidence belief is discussed later in section 3.4.

For X to have trust in Y with respect to task \( \alpha \), X must believe that Y will actually do \( \alpha \) [CAS98, CAS00, CAS01b, FAL04a, T3G]. The belief that agent Y will do a task \( \alpha \) is known as the will-do belief or the disposition belief. In the case of Y being a cognitive agent the disposition belief is supported by at least two other beliefs. X must believe that Y (a cognitive agent) has decided to execute \( \alpha \). The belief that agent Y has decided to execute \( \alpha \) is known as the willingness belief [CAS98, CAS00, CAS01b, FAL04a, T3G]. X must also believe that Y will persist to complete \( \alpha \) and not simply give up or reverse its decision to execute \( \alpha \). Agent X’s belief that agent Y will persist is known as the persistence belief [CAS98, CAS00, CAS01b, FAL04a, T3G]. Both the persistence belief and the willingness belief are in turn supported by the above mentioned self-confidence belief [T3G]. Thus the self-confidence belief supports the disposition belief.

Agent X must also believe that it needs to rely on Y meaning that X must believe that there is sufficient utility in delegating \( \alpha \) to Y. Agent X’s belief that it needs to rely on Y is known as the dependence belief [CAS98, CAS00, CAS01b, FAL04a, McL06,
The self-confidence belief mentioned above is one of two secondary beliefs that require special mention. It is important to understand how the confidence belief supports the competence belief and the disposition belief. Consider that X must believe that Y is confident about its own ability and is confident enough to execute task $\alpha$ successfully [T3G]; how can X believe in Y if Y does not believe in itself? If X believes that Y is self-confident then X may reason that there is a good probability that Y’s self-confidence is due to Y’s knowledge of its (Y’s) competence. To see why the self-confidence belief supports the disposition belief, consider the case where Y is not self-confident with respect to task $\alpha$; agent Y does not believe that it can complete $\alpha$ successfully. In the best case where Y attempts to perform $\alpha$, if it encounters some obstacle, Y will almost certainly fail to persist with the task. If agent Y was convinced of its ability to complete $\alpha$ successfully in the given environment then agent Y would be more likely to persist in the event of encountered obstacles.

The other secondary belief is a motivation belief. X must believe that Y has motives for agreeing to adopt a goal or to attempt a task $\alpha$ [CAS98, CAS01b, T3G]. Furthermore, X (being a rational agent) must believe that Y’s motives for assisting X will prevail in the case of conflict with other motives. The motivation belief, like the self confidence belief, only applies in the case of strong delegation [CAS01b]. The motivation belief is discussed further in the section on delegation.

The beliefs of X’s trust in Y are not necessarily limited to the beliefs mentioned here. There may be other secondary beliefs that apply; for example X may have certain beliefs that Y is not dangerous or malicious.

Section 3.4.2 has discussed the beliefs relating to only one form of trust – X’s trust in Y. The following paragraph discusses the beliefs relating to X’s trust in the environment.
3.4.3 Beliefs of External Trust
So far only the beliefs that form the basis of X’s trust in Y have been discussed. However, X’s trust in the environment is also important and similarly is based on various beliefs. As far as X’s trust in the environment is concerned, at least two primary beliefs can be identified; the first of the beliefs of external trust is the **opportunity belief**. The opportunity belief is X’s belief that there are sufficient resources or opportunities in the environment for the successful completion of \( \alpha \) [CAS98, CAS01b, FAL04a, T3G]. If X does not believe that sufficient resources or opportunities exist then X lacks confidence in the environment.

The other primary belief relates to the absence of threats in the environment. The belief is known as the **safety belief**. The safety belief means that X believes that the environment is safe enough for \( \alpha \) to be performed [CAS98, CAS01b, FAL04a, T3G]. The safety belief implies that agent X believes that the environment is free from threats, the consequences of which (when executing \( \alpha \)) would be so severe that X could not accept the risk. Furthermore, X must also believe that the frequency of the occurrence of any threat (other than mere nuisances) is limited so as not to make it impossible to complete \( \alpha \).

The opportunity belief and the safety belief are general beliefs and can be applied to any environment [FAL04a]. The opportunity belief and the safety belief may well be supported by other, more environment specific beliefs.

The above paragraphs discuss the beliefs of different forms of trust. It may be asked whether such a decomposition of trust into its constituent beliefs is of any relevance. The following paragraph addresses the issue of the relevance of the decomposition.

3.4.4 Importance of Decomposition
To understand why the distinction between trust in an agent (internal trust) and trust in the environment (external trust) should be made consider the situation where an agent X trusts an agent Y with respect to task \( \alpha \). Agent Y fails to successfully complete task \( \alpha \) because of environmental factors beyond its control. Thus it is possible that agent X will perceive betrayal when in fact agent Y has not intentionally
betrayed agent X. As the above example illustrates, it is important to distinguish between trust in the environment and trust in an agent or entity [FAL04a].

3.4.5 Summary
Section 3.4 introduces a decomposition of trust. The trust that a goal will be achieved can be decomposed into the trust in the agent and the trust in the environment. The trust in the agent is based on several beliefs. Similarly the trust in the environment is based on various supporting beliefs. Section 3.4 contributes to the dissertation by explaining the cognitive nature of trust. Furthermore, the section provides an indication of the beliefs that an agent must hold in order for it to trust another agent. It is obvious that the beliefs of trust play a central role in a cognitive agent’s evaluation of another agent’s trustworthiness. The following section looks at potential sources of the beliefs of trust.

3.5 Sources of Trust
3.5.1 Sources of Belief
The beliefs of the various forms of trust don’t simply exist – they stem from certain sources. Sources of belief are the topic of section 3.5.

3.5.2 Direct Experience
The most common source of belief is direct experience. When an agent (say X) is directly involved in some interaction then X gains direct experience. X may also gain direct experience through directly observing some event that does not necessarily involve X. Assuming that X is competent in perceiving what actually happens then direct experience is generally the most reliable source of trust [CAS03, FAL04a, T3G]. However, there are times when an agent’s direct experience may be insufficient [SAB02]. In cases where direct experience is insufficient other sources of belief may be useful [HUY04, TEA06]. One such example is reputation, as discussed next.

3.5.3 Reputation
Reputation is another common source of belief [CAS03]. Agent Y’s reputation refers to another agent’s opinion of Y [CAS03, SAB01]. Alternatively, reputation could refer to a group of agent’s collective opinion of Y [HUY04, SAB02]. Typically agent
X will only have access to the reputation information and will not know precisely the reasons supporting the reputation rating. Furthermore, an unfortunate consequence of relying on reputation information is that the problem of trust is extended to the agent providing the information. Can X trust agent Z’s opinion or evaluation of Y?

While experience and reputation are commonly used sources of trust, they are not the only source. The following paragraph discusses categorisation.

3.5.4 Categorisation
Beliefs can also be derived from stereotypes or more generally – categorisation [CAS03]. In a multi-agent environment, the various agents may be classified into classes according to shared attributes or properties. Any particular cognitive agent may have beliefs pertaining to some class of agents. However, classifying agents according to a single specific class or category can be misleading; the agent may not ‘fit’ the class exactly. Hence categorisation as a source of belief may be misleading.

Categorisation may be used more effectively if combined with reasoning. Reasoning as a source of trust is discussed next.

3.5.5 Reasoning
Beliefs may also be derived from reasoning [CAS03]. Agent X could, for example, believe that it and agent Y are members of the same class. X could also believe that another agent Z belongs to the same class as Y. X can then conclude, by transitivity, that it and Z are members of the same class. Therefore it is possible for an agent to reason about existing beliefs and in so doing infer new beliefs. It is important to note however, that the process of finding equivalent beliefs is strictly dependant on uncertain premises (i.e. the existing beliefs). The approach to reasoning as outlined here is based on the assumption that the existing beliefs are in fact both true and justified. Since belief implies uncertainty it may be the case that the existing beliefs are not true and justified and then use of the approach could amount to wasted effort.

3.5.6 Relationship Analysis
Another potential source of beliefs is relationship analysis as outlined in the literature [ASH05, SAB02]. Relationship Analysis can be considered as a composite source,
including categorisation and reasoning. The approach to relationship analysis as a strategy for evaluating trust is presented in chapter 5, so therefore it is only briefly mentioned in general terms here.

Relationship analysis involves identifying possible existing relationships between agents; categorising the relationships; interpreting the semantics of such relationships and then reasoning about them [ASH05]. The advantage of the relationship analysis approach is that it can often be used by an agent that is new to an environment. Such an agent has no direct experience to rely on and relying on reputation information provided by third parties raises the question of the third parties’ trustworthiness. Therefore, relationship analysis is a potential solution to the problem of initial trust evaluation as faced by an inexperienced agent. However, relationship analysis is not without its own problems as discussed in chapter 5.

3.5.7 Summary

The above section looks at sources of belief. Direct experience is the most obvious source of belief. An agent may also rely on the opinions provided by other agents, thus reputation is a source of belief. Categorisation and reasoning may also be seen as sources of belief. Relationship analysis serves as an example of both categorisation and reasoning. An agent’s beliefs of trust and any changes to those beliefs stem from the aforementioned sources of belief. Section 3.5 contributes to the dissertation by explaining how the beliefs of trust discussed in section 3.4 may arise. Section 3.5 sheds some light on how one cognitive agent may arrive at the beliefs leading to trust. The section that follows introduces the concept of delegation.

3.6 Delegation

3.6.1 What is Delegation?

Delegation is an action that results in a social relationship between two agents [CAS98]. Delegation is an action that is (usually) the result of reliance, where reliance is the decision to rely on another agent. The decision to rely on another agent is based on trust (See figure 3.2). Thus, according to Castelfranchi and Falcone, trust is the mental counterpart of delegation [CAS98, CAS01b, CAS01c, FAL04a, ESF01]. While trust is normally necessary for delegation it is not sufficient; the degree of trust
may simply not be sufficient or there may be other external factors preventing delegation [CAS01b]. It is also possible that delegation may occur without sufficient trust; this could occur due to some exceptional circumstances where perhaps the delegating agent lacks both the necessary information and alternatives. The action of delegation can be categorised into two broad categories which are now discussed.

### 3.6.2 Weak Delegation

Consider a situation where the delegating agent X trusts another agent Y sufficiently and thus has decided to rely on Y. Weak delegation occurs when X does not inform Y of the delegation [CAS98, CAS01b]. More precisely X does not communicate to Y its intention to exploit Y’s action. Generally Y is not aware of X’s decision to rely on it meaning that Y is unaware of X’s plan to exploit its action. However, it is not necessarily the case that agent Y is unaware of agent X’s plan to exploit its action [CAS98, CAS01b].

The action of weak delegation is often the action of inaction (doing nothing). Once Y has performed its action X will subsequently exploit the result.

Weak delegation is only one form of delegation. Another form of delegation is subsequently discussed.

### 3.6.3 Strong Delegation

Strong delegation occurs in the above outlined situation when a cognitive agent Y is consciously aware of X’s decision to rely on it [CAS98, CAS01b]. Furthermore, X communicates its intention to exploit Y’s action. It is important to note that strong delegation can only occur between two cognitive agents [T3G]. In the case of strong delegation, X may have some influence over Y. There may also be a contract or agreement between X and Y. Thus the action of strong delegation could include negotiating a contract. Alternatively, it could be the provoking or inducing of a desired behaviour – as in the case of coercive delegation [T3G].

Strong delegation differs from weak delegation not only in that X communicates the delegation to Y, but also because X’s mental state is different. X has at least one
belief that would not exist in the case of weak delegation [CAS01b]. The additional belief that exists in the case of strong delegation is discussed next.

3.6.4 The Motivation Belief of Strong Delegation
As was mentioned in the section on the beliefs of trust, the motivation belief only applies in the case of strong delegation [CAS01b]. To understand why X must believe in Y’s motivation consider the following: Y as a rational agent acts in a manner that it expects will maximize its performance measure. Thus Y is naturally selfish and always behaves in a manner that is consistent with achieving its own goals. Y, after adopting X’s task $\alpha$, decides that it is no longer convenient to persist in attempting to complete the delegated task. Agent Y may no longer persist to complete $\alpha$ possibly because the achievement or completion of task $\alpha$ no longer coincides with Y realising its own goal.

Given the rationality of X and Y it would seem contradictory to expect X to believe that Y’s motivation for doing $\alpha$ will prevail over another of Y’s more goal directed motives [CAS01b]. Similarly, it would seem contradictory for Y not to pursue its goals and – in so doing – continue with the delegated task. Since X and Y are not irrational agents it must be the case that Y has goals such as creating and maintaining a good reputation [CAS01b]. The reputation related goals must be important for Y and thus Y’s motivation with respect to the creation and maintenance of good reputation must prevail in the case of conflict among motives [CAS01b]. Furthermore, X must believe that agent Y has altruistic goals and that agent Y is committed to achieving its altruistic goals.

3.6.5 Summary
Section 3.6 presents the concept of delegation. Delegation can be considered as some action that follows as a consequence of reliance. Two different types of delegation can be distinguished. Weak delegation implies that the agent being relied upon is unaware of the delegation. Strong delegation implies that the agent being relied upon is aware of the delegation. The motivation belief applies in the case of strong delegation. The motivation belief implies that the agent being relied upon has some motivation to assist the agent delegating the goal.
Section 3.6 contributes by explaining the importance of trust in the context of delegation. Trust as the mental component of delegation is normally needed in order for an agent to rationally delegate a task or a goal to another agent. An understanding of delegation and the beliefs involved is necessary for a complete theory of trust.

So far in chapter 3 some basic foundations have been laid down. The following sections build on these foundations (i.e. the semantics of trust; the beliefs of trust; the sources of trust; and the concept of delegation) by dealing with some more advanced topics. The following section considers the dynamics of trust.

3.7 The Dynamics of Trust

3.7.1 Trust is Dynamic

Section 3.5 discussed possible sources of belief upon which trust is based. As the belief sources are modified, the beliefs upon which trust is based may change and thus trust changes. Section 3.6 discussed the action that results when one agent trusts another, and decides to rely on it. Delegation also affects trust. Furthermore, trust itself may influence trust. Hence trust is a dynamic concept. The dynamics of trust is the topic of section 3.7.

3.7.2 A Naïve Approach to Trust Evaluation Based on Direct Experience

Typically one agent’s evaluation of another agent’s trustworthiness increases with the success of the delegated task (when the evaluating agent, say X, is not betrayed). Similarly the evaluation decreases with the failure of the delegated task (when X is betrayed). Increasing (respectively decreasing) the level of trust based only on the outcome of the delegated task is traditionally (and intuitively) what is anticipated when it comes to re-evaluating another agent’s trustworthiness. However, changing the level of trust on the basis of the outcome alone is perhaps a somewhat naïve view [FAL04a]; especially given the decomposition of trust in section 3.4. If agent X’s trust is betrayed in relation to the successful performance of task α, then one of the following is true [FAL04a]:

- Only X’s trust in the environment is betrayed.
- Only X’s trust in the trusted agent Y is betrayed.
- Both X’s trust in the environment and in Y is betrayed.
Clearly, when an agent’s trust is betrayed, it needs to be re-evaluated. The above analysis indicates the importance of evaluating precisely the reasons for the success or failure of a delegated task. The success or failure can ultimately be attributed to certain external factors of trust (environmental factors), the internal factors of trust (factors relating to the trusted agent), or a combination of the two [FAL04a]. The attribution of the outcome is discussed subsequently.

3.7.3 The Attribution of Success or Failure
As mentioned above, the success or failure of \( \alpha \) can be attributed to internal factors, external factors or a combination of internal and external factors. However, the nature of the internal and external factors should also be taken into account [FAL04a]. Is a particular environmental factor recurring problem or just an occasional nuisance? The question is posed in the work by Falcone and Casetlfranchi [FAL04a]; the attribution of the success or failure of \( \alpha \) can be described by an intersection of a row and column (cell) in the matrix of figure 3.3.

<table>
<thead>
<tr>
<th></th>
<th>Frequently Recurring</th>
<th>Occasional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal</td>
<td>( \alpha )</td>
<td></td>
</tr>
<tr>
<td>External</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 3.3 Characterisation of an Outcome**

Given the above suggested attribution for the success or failure of \( \alpha \), it becomes possible for a cognitive agent X to reason about an increase or decrease in its trust evaluation of Y. The following decision tree (figure 3.4), adapted from the work by Falcone et al [FAL04a], illustrates the attribution reasoning process.

Note however, that in both the matrix of figure 3.3 and the decision tree of figure 3.4 failure or success cannot be attributed to a combination of internal and external factors. Therefore in the cited paper by Falcone et al a simplifying assumption is made that an agent cannot simultaneously be betrayed by the environment and the trusted agent [FAL04a]. A more thorough approach would require a more complex decision process.
Given the above suggested process it should be apparent that X’s future trust may decrease even though $\alpha$ was successfully completed. Similarly, X’s future trust may increase in the case of failure to complete $\alpha$. The attribution approach described here is quite different from the traditional approach to trust reinforcement. It also suggests that a level of reasoning is required. Therefore applying reinforcement learning methods to the problem of trust reinforcement may be an inadequate solution.

So far section 3.7 has focused only on how trust changes (or should change) with direct experience. However, trust also changes as other belief sources are modified. The changes in trust due to the modification of other belief sources tend to be more trivial. For the sake of completeness, changes due to modification of other belief sources are subsequently discussed.
3.7.4 Modification of Belief Source Impacts Trust
In the event of new reputation information, agent X needs to evaluate the credibility of the information as well as its trust in the provider of the information. Once credibility has been considered agent X may adjust its trust accordingly.

Categorisation is less likely to change unless agents have the ability to change their defining attributes. Nevertheless, it may be prudent for agent X to periodically re-evaluate its categorisation of other agents and adjust its trust accordingly.

It is also prudent for an agent to reason about its beliefs whenever they change. Consider the example in the section on reasoning as a source of trust where X infers an equivalent belief that it and Z are members of the same class. Should information come to light that results in X changing its belief that it and Y belong to the same class then the equivalent belief is no longer valid. Ultimately X must check the consistency of its beliefs from time to time.

3.7.5 Trust Influences Trust
Up until section 3.7.5, only changes in trust due to the modification of beliefs based on the modification of belief sources have been considered. However, it is possible that trust can influence trust. Firstly, trust can enhance trust [CAS01c, FAL04a, T3G]. Consider the situation where X trusts in Y for the successful execution of $\alpha$. If Y knows of X’s trust, irrespective of whether X has delegated the task, Y’s self-confidence may be increased. The degree to which Y’s self-confidence is increased is dependant on Y’s respect for X. Intuitively, Y’s self-confidence should increase depending on the strength of Y’s belief in X as an authority (the strength of Y’s belief in X’s competence). As a consequence of the supposed increase in Y’s self-confidence (and given that X’s belief about Y’s self-confidence supports X’s disposition belief), Y may be more willing to do $\alpha$ and to persist in doing $\alpha$. Therefore, if X knows of Y’s knowledge of X’s trust in Y then X’s trust in Y may increase [CAS01c, FAL04a, T3G]. Alternatively, if X is unaware of Y’s knowledge of X’s trust in Y then X’s evaluation of Y’s trustworthiness may be inaccurate [CAS01c, FAL04a, T3G].
Another way in which trust affects trust is through the spreading of trust among multiple agents [T3G]. There are two situations worthy of discussion here [CAS01c, T3G]. Consider an agent X who is regarded as an authority in the environment. Also consider that X trusts in Y for the successful completion of $\alpha$. Agents W and Z trust X as a good evaluator of trust and therefore trust in Y because of X’s authority and trust in Y [CAS01c, T3G]. A similar situation may arise in the environment where a number of agents may trust in Y. Agent X may consider Y trustworthy because of the sheer number of agents that have trust in Y [CAS01c, T3G]. However, if all of the trusting agents are incompetent evaluators of trust then ‘following the flock’ is a risky strategy.

One further case of trust’s influence on trust occurs in the transitive case where X trusts Y as a competent and honest evaluator of trust. Since X trusts Y and Y trusts Z, then X trusts Z [CAS01c, T3G].

The final case considered here is where Z has trust in W. Also X has trust in Z as an authority. Furthermore, X believes that Y is fundamentally similar to W. Then X’s trust in Y may be influenced by analogy [CAS01c, T3G].

Trust is however, only the mental component of delegation [CAS98, CAS01b, CAS01c, FAL04a]. Since trust affects trust, it would be reasonable to question if delegation affects trust. The effect of delegation on trust is the topic of the remainder of section 3.7.

3.7.6 Delegation and Reliance Influence Trust

For a complete understanding of the dynamics of trust it is also necessary to consider the effect of delegation and thus reliance on trust. First the effect of weak delegation is discussed followed by the effect of strong delegation. Finally, a more general case is considered.

Consider a situation where X decides to rely on Y and thus delegates the task $\alpha$ with no explicit agreement (weak delegation). In the case that Y becomes aware of the delegation, Y’s trustworthiness could be affected [FAL04a]. Y may object to such exploitation or may become more self-confident as discussed above. If X is aware that
Y has discovered the weak delegation, then X has the opportunity to learn the extent to which Y is spontaneously collaborative [FAL04a]. If X is unaware (lacks the knowledge) of Y’s discovery of the weak delegation then X’s evaluation of Y’s trustworthiness may be inaccurate [FAL04a]. Thus, it would be prudent for X to consider the possibility of Y being aware of the weak delegation. It then becomes possible for X to attribute a degree of belief to Y’s awareness or lack of it [FAL04a].

The above paragraph considers how weak delegation affects trust. The focus now shifts to the case of strong delegation. In many situations, strong delegation may be preferable for X. Strong delegation may be preferable because if X has an agreement with Y to complete \( \alpha \) then Y will be more committed to the completion of the task. Y’s increased commitment makes Y more trustworthy for completing \( \alpha \) as far as X is concerned [FAL04a].

In addition to the previously mentioned reason for which strong delegation is preferable, strong delegation is also advantageous for another reason [FAL04a]. By definition Y will be aware of the delegation. Since X knows of Y’s awareness of the delegation, the problem associated with weak delegation is eliminated. That is, the possibility that X’s evaluation of Y’s trust being inaccurate due to X being unaware of Y’s knowledge of the delegation is eliminated. Hence strong delegation may be preferable to weak delegation. Ultimately however, the most important lesson in section 3.7.6 is that X needs to consider how its delegation action can affect Y’s trustworthiness.

So far the focus has been on how delegation affects X’s trust in Y. However, it can also be the case that delegation makes X more trustworthy for Y. Should an agent X decide to delegate \( \alpha \) to Y, then X becomes reliant on Y at least as far as the successful completion of \( \alpha \) is concerned. Due to X’s trust in Y and reliance upon Y, X is faced with the risk of betrayal and the associated risks. Thus X is less likely to betray Y since X, being at Y’s mercy, fears retaliation [CAS01c, FAL04a, T3G]. Therefore Y may consider X to be more trustworthy for the duration of the delegation. However, agent Y’s perception of agent X as being more trustworthy only applies in the case of strong delegation and in the sub-case of weak delegation where Y is aware of the delegation.
3.7.7 Summary
Section 3.7 introduces the very important concept of attributing the outcome of a delegated goal to agent internal factors, or environmental factors or both. The section also discusses more advanced and theoretical aspects of the dynamics of trust. The potential for trust to influence trust through the self-confidence belief is discussed. The effects of both weak and strong delegation (and therefore reliance) on trust are also addressed.

The main contribution of the section is the attribution of the outcome of the delegation. The modification of belief sources and therefore the modification of the certainty of beliefs affect the evaluation of an agent’s trustworthiness. Therefore, the theory of section 3.7 plays an important role in the determination of another agent’s trustworthiness. The following section introduces the concept of an atmosphere of trust.

3.8 Towards an Atmosphere of Trust in Multi-Agent Systems
3.8.1 Desirability of Trust Atmosphere
The section on the dynamics of trust in part focused on how trust and delegation affect trust. Given the dynamics of trust, it is clear that trust relationships among agents in the environment will evolve. Ideally, the environment should evolve towards a state where an atmosphere of trust exists [CAS01c, T3G]. An atmosphere of trust is desirable because it should result in maximum cooperation and collaboration among agents in the environment. The concept of an atmosphere of trust is the subject of the following paragraphs.

3.8.2 Defining Trust Atmosphere
According to the literature an atmosphere of trust is defined as follows [T3G]: ‘Trust atmosphere is the diffuse perception (beliefs based on experience) within a group or organisation of the fact that everybody trusts everybody, or better that the majority of the members have multiple and mutual relations of trust’. The definition states that an atmosphere of trust exists when the vast majority of agents trust more than one agent in the environment and are trusted by more than one agent in the environment. Furthermore, all the agents are aware of the numerous trust relationships.
Figure 3.5 (adapted from [T3G]) indicates the difference between many trust relationships and actual trust atmosphere. The circles represent distinct agents, while a directed edge represents a trust relationship between agents. Some of the indicated relationships are bilateral.

3.8.3 Summary
Section 3.8 discusses the concept of trust atmosphere – where every agent trusts every other agent. Trust atmosphere is clearly desirable and important. However, trust atmosphere is not always achievable. Nonetheless, a trust-based solution should be applied with a view towards creating an atmosphere of trust in multi-agent systems. Creating an atmosphere of trust is likely to minimise incidents of betrayal. Therefore, the contribution of section 3.8 is in the suggestion of a potential meta-goal for trust-based solutions, irrespective of whether the goal is achieved explicitly or implicitly.
So far in chapter 3 the emphasis has been on trust. The section that follows discusses the related concept of control.

### 3.9 A Theory of Control

#### 3.9.1 Trust and Control

Section 3.9 focuses on the complex relationship between trust and control. In section 3.3 it was asserted that trust and control cannot coexist; if an agent X controls an agent Y then X does not trust Y. The assertion is however, an oversimplification of the relationship between trust and control [CAS00]. Despite the inversely proportional relationship, control can in fact build or increase trust [CAS00]. The seemingly paradoxical relationship between trust and control is subsequently discussed, but first it is important to understand precisely what is meant by control.

#### 3.9.2 What is Control?

Castelfranchi and Falcone define control as an action that [CAS00]:

a. Is aimed at determining whether another action has been successfully carried out and if a given state in the environment has been realised.

b. Is aimed at responding to deviations and unexpected events that may occur in the environment.

From the above definition it follows that control consists of monitoring and intervention [CAS00, T3G]; ‘aimed at determining’ implies monitoring whilst ‘aimed at responding’ implies intervention.

Two types of monitoring can be identified; feedback and surveillance [CAS00, T3G]. Feedback occurs when the trusted agent Y provides X with some form of progress reports. Feedback could happen on an ad-hoc basis or at agreed intervals. Feedback occurs when X delegates the control task of monitoring to Y. The delegation of the control task is discussed later in the section. Surveillance occurs when X directly observes Y’s actions and/or changes to the state of the environment.
Castelfranchi and Falcone identify five types of intervention [CAS00]. The types of intervention include:

- Interruption – the task is stopped either temporarily or permanently.
- Substitution – allocating either part of task or the whole task to another agent.
- Correction – part of the task or the whole task is changed.
- Abstraction or specification – making the task more or less constrained.
- Repairing – specification of new actions necessary to complete the task successfully.

In the event that an agent X decides in favour of controlling the execution of a delegated task by a cognitive agent Y there are at least 3 possibilities [CAS00]:

- X directly controls the execution of the task.
- X trusts Y to control the execution of the task and delegates the control task to Y.
- X delegates the control task to a third party.

It is important to note that for the first possibility to occur, X must have sufficient confidence or trust in its own ability to control Y. Alternatively, X must believe that it is capable of influencing Y, and inducing Y to believe that control exists. Should agent X lack confidence in its ability to control agent Y and believe that agent Y is beyond influence then agent X may resort to delegating control to a third party. However, delegating control to a third party requires trust in the third party [CAS98, CAS01b]. Third party trust is discussed in section 3.10 on trust in contracts. In some cases X may also resort to the second option. Delegating the control task is seemingly irrational and somewhat contradictory; X trusts Y to provide control (e.g. accurate feedback) but does not trust Y to complete task \( \alpha \) successfully without control, hence X’s perceived need for control. Surely it is a conflict of interest for Y to provide control for the task \( \alpha \)?

The question surrounding the rationality of delegating the control task is discussed further in paragraph 3.9.4, but first it is important to understand the complex relationship between trust and control.
3.9.3 Trust and Control: A Paradoxical Relationship?

It has been asserted that trust and control cannot coexist. The assertion is however, an oversimplification [CAS00]. It is true that X does not trust Y if X wants to control or monitor Y in some way. The notion of trust referred to here, is quite strict and specific; it is X’s trust in Y [CAS00]. It should be noted that agent X’s trust in agent Y is too narrow a view of trust. It is also important to consider the context in which the trust occurs.

If X trusts Y to complete task α (X has trust in Y) it means that X believes that Y is competent with respect to task α. X also believes that Y will do task α. In other words, X has both a competence belief about Y and a disposition belief about Y with respect to α. Both the competence belief and the disposition belief are constituents of X’s trust in Y, as discussed in the section on the beliefs of trust. However, X’s trust in Y is only a part of the total trust required in order to delegate α to Y [CAS00, FAL04a, T3G]; X trusts that α will be achieved and that ultimately some goal g will be achieved. The trust that α will be accomplished or that g will be achieved is relative to α and is a broader form of trust [CAS00, FAL04a, T3G].

Given the broader form of trust it is possible that control and trust can coexist [CAS00]. Indeed, X may lack the trust in Y (a more specific form of trust), but due to external factors may still trust that α will be successfully accomplished. Therefore, despite the lack of trust in Y’s ability and/or willingness, X trusts that α will be successfully accomplished due to sufficient trust in the environment in which α is to be performed. Agent X’s lack of trust in Y may give rise to a desire for X to have some control [CAS00]. The control that agent X desires is aimed at increasing the trust that α will be accomplished or that g will be achieved. Figure 3.6 illustrates the idea.

With the relationship between trust and control (partly) explained, it becomes possible to discuss the rationality of X delegating control to Y as discussed in section 3.9.2.
3.9.4 Conflict of Interest: Delegating the Control Task

Consider the scenario where X delegates the task \( \alpha \) to Y and also delegates the control of that task to Y. Section 3.9.2 questioned the rationality of such an action; is it wise for X to delegate control to Y? Surely Y cannot objectively execute the control task \( \beta \), when \( \beta \) relates to the control of task \( \alpha \)? Section 3.9.4 attempts to answer the questions posed.

Firstly, it is worth pointing out again that X’s trust in Y is relative to the task [CAS98, CAS00]. Therefore, if X distrusts Y to complete \( \alpha \) successfully (and hence wants to control Y) it does not necessarily mean that X distrusts Y to complete \( \beta \) successfully. Clearly X has doubts about Y’s competence or willingness with respect to Y’s performance of \( \alpha \). However, X must still believe that \( \alpha \) will be achieved (perhaps due
to environmental factors). If X does not believe that $\alpha$ will be achieved then X would generally not delegate the task $\alpha$ to Y.

In order for X to delegate the control task $\beta$ to Y, X must believe in Y’s competence and willingness to complete $\beta$ [CAS98, CAS00]. Also, X must believe that $\beta$ will be successfully completed. It would be irrational for X to delegate $\beta$ to Y if X does not trust that $\beta$ will be completed successfully. Similarly, if X does not trust in Y to perform $\beta$ then delegation of $\beta$ to Y is irrational. Should X lack trust in Y to perform $\beta$ (irrespective of X’s trust or distrust that $\beta$ will be completed successfully) X will feel the need to control Y’s performance of $\beta$ [CAS00]. In such an eventuality X desires to control the control tasks. Controlling the control tasks is reminiscent of infinite recursion and it is most certainly better for X not to delegate the control task to Y.

Section 3.9.4 discusses why it may be rational for X to delegate the control task to Y. The following section discusses two types of control.

**3.9.5 Types of Control**

Section 3.9.5 summarises two types of control as identified by Castelfranchi and Falcone [CAS00]. More importantly the section aims to show how the two forms of control could influence trust. As mentioned, it is possible to distinguish between two primary types of control [CAS00, T3G]:

- Influencing control.
- Adjustment control.

Each form of control and how it influences trust is discussed in subsequent paragraphs.

The fundamental purpose of influencing control is to prevent violations or mistakes [CAS00]. As has already been discussed in section 3.9.2, control can increase one form of trust – the ‘global trust’. However, influencing control can in fact increase another form of trust – X’s trust in Y [CAS00]. It seems as though a contradiction exists; where there is control there can be no trust. However, this is a very restricted binary view of one agent’s trust in another. To avoid a contradiction, it is necessary to adopt a ‘degrees of trust’ approach [CAS98, CAS00, CAS01b, T3G]. To understand why control could possibly increase X’s trust in Y, consider the following case.
Agent Y may seek or desire states where it has a good reputation. That is Y has a higher utility for states in which its reputation is better. Thus Y may be concerned about X’s beliefs relating to Y’s competence (assuming competence affects Y’s reputation). As a consequence, Y may take more care in performing task \( \alpha \) if Y knows that it is being monitored directly or indirectly by X [CAS00]. Similarly Y is more likely to do the task. Should X believe this to be the case (i.e. should X believe that Y is concerned about its reputation) then X will take the belief into account in evaluating Y’s trustworthiness [CAS00]. Therefore X’s trust in Y may be higher with control.

Clearly, the influencing type of control is essentially the act of monitoring alone [CAS00]. Also, for the influencing type control to result in an increase in X’s trust in Y, it requires that Y knows that it is being monitored [CAS00]. Furthermore any increase in X’s trust in Y due to monitoring is based on the assumption that Y actually cares about its reputation.

Adjustment control differs from influencing control in that adjustment control aims to mitigate the risks that may occur as a consequence of Y’s mistakes [CAS00]. However, Castelfranchi and Falcone argue that the adjustment form of control can also prevent violations [CAS00]; violations are prevented when Y (the controlled agent) is aware of the control and cares about its reputation. The adjustment form of control implies monitoring and intervention [CAS00]. Therefore adjustment control increases X’s trust that \( \alpha \) will be completed successfully because in general, X believes that there is a greater probability of completing the task successfully with the possibility of intervention.

### 3.9.6 Summary

Section 3.9 introduced control. Section 3.9.2 defines control and discusses two types of monitoring (feedback and surveillance). Five types of intervention are also listed. Section 3.9.3 discussed how control can actually increase one form of trust (trust that \( \alpha \) will be achieved) while potentially eliminating another form of trust (X’s trust in Y). Section 3.9.4 discusses how and when it can be a rational action for X to delegate the control task \( \beta \) as well as the original task \( \alpha \) to Y. Section 3.9.5 distinguishes between two types of control (influencing control and adjustment control) as
identified by Castelfranchi and Falcone [CAS00]. Additionally, it is also shown how influencing control can potentially increase the more strict notion of trust – X’s trust in Y.

Section 3.9 contributes by providing a theory of control that is important for a complete understanding of the theory of trust. Therefore, while the contribution may seem ancillary it is nonetheless necessary. Furthermore, trust and control forms an important part of any contractual relationship. The role of control is highlighted in the following section on third party trust.

3.10 Third Party Trust: Trust in Contracts

3.10.1 A Shift in the Focus of Trust

Again the notion of trust and contracts seems somewhat contradictory. Contracts are put in place precisely because parties do not trust each other and thus want to be protected by a contract [CAS98, CAS01b]. So surely if agent X wants to be protected by a contract before delegating task \( \alpha \) to agent Y, then X does not trust Y? Again, an oversimplification exists. While it is certainly true that X lacks trust in Y with respect to the performance of \( \alpha \), it is also the case that there is a shift in focus of X’s trust [CAS98, CAS01b].

3.10.2 The Beliefs of Contractual Trust

The relationships between the beliefs of contractual trust are conceptualised in figure 3.7. X believes that the contract offers it protection, meaning that X has the belief that Y will do what is promised in the contract [CAS98, CAS01b]. The ‘will-do’ belief or disposition belief must be based on the belief that Y is a ‘moral’ or ‘ethical’ agent or the belief that Y fears the consequences of violating the contract [CAS01b]. Ultimately, X clearly trusts that y will not violate the contract and thus Y will do as promised in the contract.
Should X believe that a contract offers protection without knowledge (or evidence) of the morality of Y then X must believe that Y fears the consequences of violating the contract [CAS01b]. For the protection belief to be warranted there are two requirements:

- The contract must afford adequate protection for both parties (i.e. the consequences of violation must be severe enough).
- Their must be an arbiter to assess contract violations and then punish accordingly.

The authority that assesses contract violation is another cognitive agent – A. It is critical that Y believes that A is capable of accurately assessing violations and then imposing sanction or punishment [CAS01b]. The authority should also be an impartial arbiter to avoid any conflict of interest. The existence of an impartial arbiter is fundamental to the success of such a scheme.

3.10.3 A Three Party Relationship

The introduction of an authority results in the following tripartite relationships [CAS98, CAS01b]: The client (X) trusts the authority (A) and relies on its ability to control the contractor (Y). The client also trusts the contractor with respect to the performance of the contracted task because of the contractor’s ethical values and/or because of the contractor’s respect for the authority and the fear of sanction. The
contractor on the other hand, trusts the authority to be fair. Furthermore, assuming the above mentioned requirements are met, the contractor will respect the authority and will fear sanction or punishment.

3.10.4 A Note on the Client’s Trust in the Contractor
It is important to note that while X’s trust in Y may not be enough (thus a contract is required), X must have a degree of trust in Y; X must have a degree of belief in the competence and willingness of Y (otherwise X would not delegate to Y at all).

3.10.5 Summary
Section 3.10 summarises the theory of contractual trust. Contractual trust involves an arbiter or authority in addition to the client and the contractor [CAS98, CAS01b]. The contract must offer protection to both parties and the authority must be able to assess and punish violations of the contract. The introduction of a contract and an authority stems from a lack of trust [CAS98, CAS01b]. However, the use of contracts implies the existence of several beliefs of contractual trust and therefore a shift in the focus of trust.

The work presented in section 3.10 is based on extensive theory of the preceding sections. The inclusion of the topic of contractual trust is important in illustrating how the notions of trust and control compliment one another. Therefore, including a section on contractual trust is necessary for completeness sake. The following section concludes the chapter.

3.11 Conclusion
Chapter 3 summarises the extensive trust theory as found in the literature [ASH05, CAS98, CAS00, CAS01a, CAS01b, CAS01c, CAS03, ESF01, FAL04a, FAL04b, HUY04, McL06, SAB01, SAB02, TEA06, T3G]. The chapter focuses on the concept of trust by defining trust and discussing the semantics of trust. A decomposition of trust into its constituent beliefs is provided. Various sources of trust are introduced. The concept of delegation is also discussed. The attribution of the outcome of a delegated goal is presented, building on the previous topics.
More advanced topics are also covered. Attention is paid to the dynamics of trust and how trust, reliance and delegation can influence trust. The relationship between trust and control also receives thorough attention. Finally, the useful notion of contractual trust is presented.

The concepts presented in chapter 3 form an important foundation on which to base a solution. The theory summarised in chapter 3 is the first step towards solving the problems as outlined in the first chapter. While many of the concepts are both useful and necessary, others are quite advanced and theoretical. The more theoretical concepts (for example, the effect of trust, reliance and delegation on trust) are only suitable when dealing with highly advanced agents with advanced cognitive and psychological abilities. The more advanced and theoretical concepts are nonetheless necessary for a complete theory of trust. A complete theory of trust (no matter how theoretical and advanced) is advantageous given the complexity of the problem outlined in chapter 1.

Chapter 3 also contributes by shedding some light on how an agent might reach a rational evaluation of another agent’s trustworthiness. In particular, chapter 3 addresses what it means to trust and what trust and deciding to trust involves. The research presented in chapter 3 thoroughly investigates the semantics of trust.

After having presented an extensive theory of trust, attention can now be given to models and methods suitable for the evaluation of trustworthiness. The next chapter addresses the topic of models and methods developed for the purposes of trust evaluation.
Chapter 4
Trust in Multi-Agent Systems: The State of the Art

4.1 Introduction
Chapter 3 provided a theoretical foundation necessary for an analytical approach to studying the different trust models and research contributions in the literature. The application of trust theory will be evident in some (not all) models and contributions that are presented in chapter 4. Chapter 4 provides an overview of some of the models that have been presented as solutions to the problem of trust evaluation. Furthermore, the models that are the topic of chapter 4 are analysed according to the criteria presented later in chapter 4. It is expected that through surveying and analysing models aimed at agent-oriented trust evaluation important techniques can be identified and later applied to the problem set out in the chapter 1.

The chapter is arranged as follows. Section 4.2 discusses the fuzzy cognitive map approach to trust evaluation – an approach that follows directly from the theory of chapter 3. Section 4.3 looks at a more statistical approach to trust evaluation. Section 4.4 presents TRAVOS, a more probabilistic approach to trust evaluation. Section 4.5 provides an overview of REGRET, a reputation focused approach. Section 4.6 presents the FIRE model for trust evaluation. Section 4.7 provides criteria for evaluating trust evaluation models. Subsequently, each of the above mentioned models is evaluated. A tabular comparison is also provided. Finally, section 4.8 concludes the chapter.

4.2 The Fuzzy Cognitive Map Approach
4.2.1 Overview
Fuzzy approaches to trust evaluation are useful as they allow for the implementation of degrees of trustworthiness [CAS03, LAM03, REH05]. A fuzzy approach stemming from cognitive trust theory (chapter 3) serves as the basis of the trust model presented by Castelfranchi, Falcone, and Pezzulo [CAS03]. At the heart of the implementation is the application of fuzzy cognitive maps (FCMs). The use of an FCM in solving the trust evaluation problem is the topic of section 4.2.
4.2.2 What is an FCM?

‘An FCM is an additive fuzzy system with feedback’ [CAS03]. The FCM is a tree-like, directed graph structure. The nodes represent contributory concepts while the edges represent the power or the effect of one node over another connected node. Initial values for leaf nodes are set manually [CAS03, FAL04b]. The initial values then propagate throughout the graph. For example consider the FCM in figure 4.1: the value of a non-leaf node is calculated by summing the products of each node value and the corresponding edge weight for all nodes connected to the node in question by an incoming edge.

![Figure 4.1 An Example FCM (adapted from [CAS03])](image)

The above paragraph describes what an FCM is. The following paragraph discusses the developers’ motivations for using FCMs.

4.2.3 Why Use FCMs?

The FCM approach is based on fuzzy logic. The fuzzy approach is particularly useful in the context of the research presented in the literature [CAS03]. Trust is difficult to estimate experimentally and the fuzzy approach makes it possible to work with intervals rather than exact values. Intervals can be assigned terms in natural language making it more intuitive to work with intervals and their associated natural language terms [CAS03].
Furthermore, it is claimed that FCMs are ‘well suited for representing a dynamic system with cause-effect relations’ [T3G]. The quote represents an intuitive view given how the values of nodes propagate through the network as mentioned above. It is also easy to see the applicability of the FCM approach to the problem of trust evaluation; trust is based on various beliefs, many of which are supported by other beliefs, all of which are based on belief sources. The assertion that the quantitative analysis of trust is based on the quantitative analysis of its cognitive constituents is consistent with the previous observation [CAS03]. Additionally, the potential support offered for feedback\(^2\) is also important given the dynamics of trust discussed in chapter 3.

It is now evident that FCMs are a useful structure suited to the problem of trust evaluation. The paragraphs that follow demonstrate the potential application of FCMs.

### 4.2.4 Using FCMs for Trust Evaluation

The implementation of a trust model utilizing the FCM approach is now discussed. For the purposes of the research presented by Castelfranchi et al only a few basic beliefs were considered [CAS03]. The beliefs considered were partitioned into beliefs about the agent in question (trustee) and beliefs pertaining to the environment. The beliefs relating to the former include a competence belief, disposition belief and a safety belief. The beliefs relating to the latter (environment) include an opportunity belief and a danger belief. The beliefs of trust considered were based on four belief sources including direct experience, reputation, categorisation and reasoning [CAS03]. Beliefs and belief sources were represented by nodes in the FCM [CAS03]. Any belief node may be connected to multiple, distinct nodes corresponding to belief sources. A conceptual view of an FCM is depicted in figure 4.2 (adapted from [CAS03]).

Each node has a numerical value that represents the value of the content of the belief or the belief source (see figure 4.1). The values of the leaf-nodes corresponding to belief sources are set manually and require human input [CAS03].

\(^2\) At present, the researchers do not consider how trust impacts beliefs; that is they do not consider feedback.
The values of all the edges are also assigned manually. The edge weights represent the impact that one linked concept has on another. Thus it is possible to specify different quantitative contributions for each higher order belief to the global trust value [CAS03]. The values or edge weights for edges between leaf-nodes (representing belief sources) and beliefs can be derived from a modulation of various factors. These factors (or meta-beliefs) include the following [CAS03]:

- Who/what the source is and the certainty about the source’s expression.
- The source’s subjective certainty about the belief.
- The credibility/trustworthiness of the source.

Once the topology of the FCM has been decided, the influence of the assigned values propagates throughout the graph. The influence of the assigned values propagates as described in the section 4.2.2. That is, for each node the sum of all the contributions is computed. Additionally for each node, the sum is then ‘squashed’ using a threshold function [CAS03]. When a stable state has been reached, the value for trustworthiness (a node in the graph) will have been computed. When the value for trustworthiness has been computed it becomes possible to make a decision. Note however, that the FCM does not take into account all the necessary factors to reach a decision. The FCM is used merely to determine a level of trust that forms part of the decision [CAS03]. There may well be other factors that need to be considered such as cost.
4.3 The Information Trust Model

4.3.1 The Model in Context

The FCM model presented above is primarily based on the theory of chapter 3. The information-based decision model (hereafter referred to as the information trust model) which is the topic of section 4.3 diverges from cognitive trust theory by taking a more statistical approach [SIE05]. The divergence from the cognitive trust theory is particularly apparent from the definition of trust found in the literature [SIE05]: ‘Trust is therefore a measure of expected deviations of behaviour along a given dimension’. The definition of trust is understandable given the context of the work in dealing with negotiating agents ([SIE05, DEB06]). The information trust model is grounded on information theory as opposed to the cognitive trust theory of chapter 3. The essence of information theory is briefly discussed in the paragraph that follows.

A negotiation process is considered to be an information exchange process [SIE05]. Certain information is communicated explicitly. For example one agent may transmit preference information to another. However, there is also a certain amount of implicit information communicated. When an agent accepts or rejects a deal, the agent reveals what it considers acceptable [SIE05]. If the same agent communicates information during the negotiation that is intended to influence its opponent’s preferences then again, the agent reveals potentially useful information. The agent may reveal its beliefs about its opponent, its opponent’s knowledge, and what it considers relevant in order to secure a deal [SIE05]. It is clear that a fair amount of potentially useful information can be gleaned from the negotiation process. However, amongst all the potentially useful information may be false and misleading information because one agent may deliberately try to deceive its counterpart in order to achieve a more desirable deal. Due to the uncertainty involved, an agent needs to attach a degree of belief to any information it uses.

The value of information communicated from agent Y to agent X is the decrease in uncertainty of X’s model of Y [SIE05]. Similarly, when X communicates information to Y, X estimates the value of that information as the decrease in uncertainty of Y’s model of X [SIE05]. The decrease in uncertainty is counteracted by an increase in uncertainty as the integrity of information decays as time expires [SIE05]. The
information trust model employs Shannon information (or negative entropy\(^3\)) as a measure of uncertainty [SIE05].

It is with the information theory background in mind that Sierra and Debenham developed the information trust model subsequently discussed [SIE05]. The information trust model is based on a so-called ‘information principle’. The information principle states that in the absence of observed facts, an agent does not speculate as to what those facts might be [SIE05]. Thus, at the heart of the information-based trust approach is an acceptance that in the presence of uncertainty it is better to focus on, and reason about what is known (rather than deal with uncertain estimates).

The above paragraphs emphasise that the information trust model is based on information theory. The following paragraph attempts to provide a high level indication as to how trust is modelled in the context of the information trust model, while the penultimate paragraph discusses the architecture of the information trust model.

**4.3.2 How Trust is modelled?**

In the context of the information trust model trust is modelled in two ways [SIE05].

The first approach is to use conditional entropy. The conditional entropy approach is based on the following principle: the fewer the number of observations that deviate from the expected observations the higher the value of trust (the more the value tends towards 1). The conditional entropy approach is perhaps more useful when any deviation from the agreed contract is undesirable [SIE05].

Another approach is to model trust as the relative entropy between the probability distribution of acceptance and the probability distribution of observation of contract execution. The relative entropy approach may be more helpful in deciding between agents.

---

\(^3\) Entropy is a measure of uncertainty in a probability distribution for a discrete random variable.
4.3.3 Using an Information-Based Trust Model

The information trust model specifies an agent architecture for negotiating agents [SIE05, DEB06]. A representation of the information trust model architecture, adapted from the literature is presented in figure 4.3 [SIE05].

![Figure 4.3 The Information Trust Model (adapted from [SIE05])](image)

When an agent X enters into a negotiation with agent Y at a specific point in time, X can start by importing messages from its inbox into an information repository using import rules [SIE05]. Integrity decay functions are then applied to each item of information [SIE05]. The agent’s knowledge base and belief base may then be
instantiated. Any inconsistencies are resolved before applying entropy-based inference (maximum entropy inference) to derive probabilities for that which remains unknown [SIE05]. The calculated probability distributions serve as important input for the negotiation strategy.

Instead of following the entire process each time (which leads to belief revision difficulties), the probability distributions may be updated in several ways [SIE05].

The distributions may be updated based on decay and the frequency of new experience. What this means is that the distribution is adjusted to reflect the decay in the integrity of information over time. Furthermore, the adjustment of the distribution due to the decay in the integrity of the information over time is offset by the reduction in uncertainty as new information comes to light due to new experiences [SIE05].

The distributions may also be affected by one agent’s knowledge of another’s preferences. If X learns that Y prefers deal $\delta$ to deal $\gamma$ and knows that Y will accept deal $\gamma$, then X can infer that Y will accept deal $\delta$. Thus the distribution can be updated by preference information [SIE05].

Furthermore, new social information may have an impact on the distribution [SIE05]. Generally, social roles adopted by agents can indicate a bias, given the rationality of agents. Thus the information trust model offers support for both the use of reputation and limited relationship analysis.

4.4 TRAVOS

4.4.1 Overview

TRAVOS (Trust and Reputation model for Agent-based Virtual Organisation$S$) arose from the need to develop a model that would ‘ensure good interactions among software agents in large scale open systems’ [TEA06]. Clearly, TRAVOS was developed to address a very difficult problem. Interestingly the TRAVOS model, while not explicitly based on the cognitive trust theory of chapter 3, is based on principles that are in part consistent with cognitive trust theory. For example TRAVOS relies on reputation to augment direct experience when there is a lack of
direct experience [TEA06]. Both direct experience and reputation are sources of trust as discussed in chapter 3. Furthermore, the TRAVOS model also adopts a ‘degrees of trust’ approach.

TRAVOS makes use of probability theory to model one agent’s trust in another [TEA06]. A probability value is useful for representing a degree or level of trust. The use of a probability value goes some way to meeting a design requirement of the model [TEA06]: the provision of a trust metric that allows a comparison of the level of trust between agents.

Another design requirement requires that TRAVOS model allows for the expression of an agent’s confidence in its level of trust toward another agent [TEA06]. The expression of confidence allows the evaluating agent to determine how much influence a particular trust metric has on the final decision to delegate a task.

One final requirement is that the TRAVOS model allows agents to take into account the credibility of reputation providers [TEA06]. In other words, agents should be able to take into account the past reliability of a reputation provider when evaluating another agent’s reputation.

As was implied in the first paragraph of section 4.4, TRAVOS allows for trust to be evaluated based on direct experience and reputation. The following section briefly outlines how trust and confidence is modelled. Section 4.4.3 deals with the modelling and the use of reputation. The last section briefly addresses how inaccurate reputation information is handled by the TRAVOS model.

4.4.2 TRAVOS: How Trust and Confidence are modelled

The TRAVOS model takes a probabilistic approach to modelling trust [TEA06]. Firstly, an agent both successfully completes a task and thus fulfils its obligations or it does not [TEA06]. The fulfilment of an obligation is denoted by the Boolean random variable $O$ as follows:

$$O = \begin{cases} 
1 & \text{If obligation fulfilled} \\
0 & \text{If obligation unfulfilled} 
\end{cases}$$
The above equality describes the outcome of a single delegated task. The set of all $t$ outcomes is denoted by $O^{1:t}$.

The unknown probability of a successful outcome ($O = 1$) is denoted by $\beta = pr(O = 1)$, $\beta \in [0..1]$. When determining a degree of trust, $\beta$ is then treated as a continuous random variable that takes on values defined by a beta probability density function\footnote{In Bayesian analysis, the beta family of probability density functions is sometimes used as an unconditional distribution for continuous random variables taking on values in the interval [0..1] [TEA06].} [TEA06, HUY06]. If a geometric interpretation of a beta probability density function (see figure 4.4) is considered then the shape of the curve is controlled by two parameters $x$ and $y$. Initially both parameters are set to 1, and thus all outcomes are equally likely [TEA06]. However, as the evaluating agent gains experience, the values of the parameters is modified [TEA06]; for every successful outcome $x$ is incremented and for every unsuccessful outcome $y$ is incremented and so the shape of the curve changes. The shape of the curve represents the degree of uncertainty associated with the possible values of $\beta$ [TEA06]. Clearly the maximum of the curve is where the degree of uncertainty is the least and thus it should be clear that the most probable value of $\beta$ occurs here. Therefore the curve maximum is the value of the degree of trust.

![Figure 4.4 An Example Beta Probability Density Function](image)

In Bayesian analysis, the beta family of probability density functions is sometimes used as an unconditional distribution for continuous random variables taking on values in the interval [0..1] [TEA06].
The degree of trust at a point in time is defined as the expected value of the probability that the agent being evaluated will fulfil its obligations (i.e. expected value of the unknown probability of a successful outcome), given the knowledge of the evaluating agent [TEA06]. The degree of trust is denoted by $\tau = E[\beta | O^1]$ . In the context of TRAVOS, the knowledge referred to above is the evaluating agent’s knowledge of previous relevant outcomes. Previous relevant outcomes are the binary outcomes of similar tasks delegated to the agent being evaluated. Dissimilar tasks are not considered relevant since an agent may behave differently depending on the task [TEA06].

The degree or level of trust $\tau$ is on its own, a number. The degree of trust does not provide an indication of the quality of the experience nor the amount of direct experience that it is based on. Therefore the TRAVOS model introduces a confidence metric, the aim of which is to allow for an agent’s confidence in its trust evaluation to be measured [TEA06]. The confidence metric $\gamma$ is defined as the conditional probability that the actual value of $\beta$ lies within a small margin of error $\varepsilon$ around $\tau$ [TEA06]. Again, if a geometric interpretation is considered then confidence can be interpreted as the proportion of the distribution (curve) that lies between $\tau - \varepsilon$ (curve maximum less the margin of error) and $\tau + \varepsilon$ (curve maximum plus the margin of error).

4.4.3 TRAVOS: How Reputation is modelled

In the above section a confidence metric $\gamma$ was discussed. If the value of $\gamma$ is below a predetermined threshold then the evaluating agent does not have enough confidence in its trust evaluation based solely on direct experience [TEA06]. Only when the value of $\gamma$ is below a predetermined threshold can the evaluating agent then rely on reputation information provided by other agents. When the evaluating agent has significant direct experience then the benefits of acquiring reputation information for evaluating trust are outweighed by the risks associated with doing so [TEA06]. The benefits may include a slight increase in the accuracy of the trust evaluation. The risks include misleading opinions and wasted effort should there be no significant change to the degree of trust [TEA06].
In the event that the value of \( \gamma \) is below the predetermined threshold the evaluating agent may start to gather reputation information until it has gathered all the reputation information available or until the value of \( \gamma \) is above the predetermined threshold – whichever occurs first [TEA06]. TRAVOS distinguishes between true reputation information and reported reputation information since some agents may be deceiving. Reported reputation information at a specific point in time is modelled as a vector consisting of the number of successful outcomes and the number of unsuccessful outcomes [TEA06].

Given multiple reputation reports, reputation information has to be combined to arrive at a degree of trust. Based on the assumption that reported reputation information is independent (based on the provider’s direct experience only) one way to combine such information is to sum the successful outcomes and unsuccessful outcomes respectively from all reputation reports received. The number of directly observed successful outcomes (respectively unsuccessful outcomes) can be added to the sum of successful outcomes (respectively sum of unsuccessful outcomes). The sum of successful outcomes can be used as the basis of parameter \( x \) and the sum of the unsuccessful outcomes can be used as the basis of parameter \( y \) [TEA06]. The two parameters \( x \) and \( y \) define the shape of the beta curve (see the previous section) from which the degree of trust is derived.

In the context of the TRAVOS model the additive approach is favoured because the resulting degree of trust is the same as it would be if all the observed outcomes had been directly observed by the recipient of the reputation information – assuming the following assumptions hold [TEA06]:

- The method of evaluating trust is both reliable and consistent among all agents.
- The behaviour of the agent being evaluated is consistent irrespective of the agent it is interacting with.
- The reputation provider provides truthful and accurate reputation information.

At least one of the above assumptions will fail to hold in more difficult and realistic cases. Thus there must be a means to deal with inaccurate and misleading reputation information. The designers of TRAVOS recognise the need to handle inaccurate reputation information and required their trust model to cater for it. The handling of
inaccurate reputation information is discussed next.

4.4.4 Handling Inaccurate and Misleading Reputation Information

The case for handling inaccurate and misleading reputation information was made above. Clearly if the integrity of reputation information is in question then it should be ignored or at least the influence on the trust evaluation should be limited. Two main approaches to limiting the influence of reputation information of questionable integrity have been identified [TEA06]. One type of approach relies on statistical methods. The other type of approach is typically based on information about the reputation provider, such as its relationship with other agents and its own reputation with regards to providing reputation information. The TRAVOS model relies on information about the reputation provider to aid in the handling of inaccurate reputation information [TEA06].

An agent evaluates the credibility of a reputation provider based on the subjective accuracy of past reputation information provided by the provider [TEA06]. Evaluating credibility involves calculating the conditional probability that the reputation provider will provide accurate information given past reputation information supplied by the reputation provider, and the subsequent outcomes as observed by the evaluator [TEA06, ZHA06]. The credibility value is then used in adjusting the reputation provider’s information or opinion [ZHA06]. The idea behind using the credibility to adjust the reputation information is that the greater the evaluating agent’s degree of belief that the reputation information received is inaccurate, the more the reputation information is adjusted to reflect equally probable outcomes. Precisely how this is achieved is beyond the scope of the dissertation. However, it should be clear that in theory, by adjusting reputation information on the basis of credibility and then aggregating the adjusted reputation information (as described in the previous section) will result in inaccurate information having less influence in the overall trust evaluation.
4.5 REGRET

4.5.1 Overview
The REGRET system is a reputation focused approach that can be employed by an agent for the purposes of trust evaluation. Sabater and Sierra define reputation as the ‘opinion or view of one (agent) about something’ [SAB01, MUI02]. Reputation (an opinion) is formed over time through interacting with other agents. For each outcome of an interaction, an impression is recorded in an impression’s database [SAB01]. Each impression is indicative of the value of an interaction with another agent. Thus reputation is decomposed into a set of impressions that the reputation is based on.

On the conceptual level reputation is seen as consisting of an individual dimension, a social dimension and an ontological dimension [SAB01, MUI02]. Each dimension as addressed in REGRET is subsequently discussed.

4.5.2 REGRET Components
The individual dimension of reputation refers to reputation that comes about due to direct interaction with another agent. Individual reputation is computed by calculating the weighted average of the appropriate values of all the relevant impressions in the impressions’ database [SAB01]. More recent impressions are considered to have more relevance and thus have a higher weight. The developers of REGRET also thought it necessary to calculate a reliability value – the purpose of which is to indicate confidence in the reputation value [SAB01]. The reliability measure is based on the number of impressions as well as the consistency of the impressions. If there is a great disparity between the values of distinct impressions a lower confidence in the reputation value results.

In a society it is possible to categorise individuals into groups. Categorisation also applies to agent societies [T3G]. Agents may be categorised into a particular group because of some shared attributes. Therefore agents belonging to the same class or group may behave in a similar manner in the same situation. The REGRET model has a social dimension of reputation that is based on the above hypothesis [SAB01, MUI02]. Therefore an agent belonging to some class or group inherits the reputation of the class or group as a whole.
The REGRET system caters for three social reputation metrics [SAB01]. The metrics include the reputations of other agents belonging to the same class as the agent being evaluated. Additionally, the reputation of the agent being evaluated according to its peers (members of the same class) needs to be measured. Finally, the evaluating agent also takes into account the opinions of its peers (agents belonging to the same class as the evaluating agent) regarding the class of agents to which the agent being evaluated belongs.

As with individual reputation, reliability or confidence measures are computed for each metric [SAB01]. Social reputation is combined additively with individual reputation.

The ontological component of REGRET exploits categorisation as a source of trust (see chapter 3). The ontological component of REGRET allows for the reputations of different concepts (defined by an ontology) to be combined [SAB01, MUI02, BIL07]. Reputation for a parent concept can be determined through combining the reputations of the parent concept’s children concepts. The reputations of the atomic concepts in the ontology (the concepts located at the leaf nodes) are determined by combining the individual and social reputations discussed above [SAB01]. The concepts are used to classify agents. Therefore, the more general the class of agent, the more reputations will have to be combined.

4.6 FIRE
4.6.1 Overview
Similarly to TRAVOS, the FIRE trust model attempts to address the challenges of open multi-agent environments [HUY04, BIL07]. The challenges posed by open multi-agent environments stem from the nature of open environments [HUY04]; different agents owned by different parties are able to join and leave the environment autonomously.

The fact that different agents representing different parties participate in open multi-agent environments means that the agents are likely to be interested in the achievement of their own goals for the benefit of their owners only [HUY04]. The
self-interested nature of the participating agents also makes it likely that competitive relationships will exist [RUS03]. As a consequence of the general competitive nature of many open multi-agent environments, the participating agents cannot be considered trustworthy by default.

Furthermore, the distinct and varied ownership of the agents coupled with their ability to autonomously join and leave the environment makes centralised control of (particularly large) open multi-agent environments very difficult [HUY04].

Moreover, given the dynamics of the system arising from the autonomy of the agents it is unlikely that any agent will have complete knowledge of its environment [HUY04]. Therefore, assuming the partial visibility of an agent’s environment, the agent will be forced to interact with other agents [JEN01].

The interactions with other self-interested agents in a partially observable environment certainly require a level of trust [HUY04]. It is with the background outlined here that FIRE was developed in accordance with the following requirements [HUY04]:

- FIRE should use multiple sources of trust; in large open multi-agent systems the availability of some sources (reputation for example) is a concern.
- FIRE should be decentralised; each agent should be able to evaluate trust for itself as centralised control is very difficult.
- FIRE should be resistant to deceitful agents; agents are likely to be deceitful given their self-interested nature.

Precisely how FIRE addresses these requirements is discussed in the following section.

### 4.6.2 FIRE Components

The FIRE trust model relies on multiple sources of trust [HUY04, HUY06, BIL07]. The sources of trust include interaction trust (direct experience), role-based trust (relationship analysis), witness reputation (reputation), and certified reputation (reputation). Each source of trust is addressed by a component of the model as is discussed in the following paragraphs.
The so-called interaction trust is trust that is fostered from direct interaction between two agents [HUY04]. The FIRE trust model borrows from the REGRET trust model by ‘exploiting the direct trust component of REGRET’ [HUY04]. The agent records its perception of each interaction with another agent (with respect to a specified attribute) as a rating in a ratings database [HUY04, SAB01]. When evaluating trustworthiness, the relevant ratings are retrieved and a value for interaction trust is calculated using a weighted average [HUY04]. The interaction trust value also has an associated reliability value. The reliability value is calculated based on the number of ratings (the more the better) and on the variability between ratings (the less the better) [HUY04, BIL07].

Role-based trust is derived from identifying the existing relationships among agents. A rule-based approach is adopted in establishing values for role-based trust [HUY04]. At a high-level, each rule specifies the expected performance of the non-evaluating agent with respect to a specified attribute, where the non-evaluating agent plays a defined role in a relationship with the evaluating agent [HUY04]. Additionally, the influence of each rule on the overall role-based trust evaluation is specified [HUY04]. Each evaluating agent has its own local rule database, to which rules may be added. In evaluating role-based trust, the relevant rules are retrieved from the database and their effect combined [HUY04]. Again, an associated reliability value is also computed using the influences of the relevant rules [HUY04].

FIRE also employs witness reputation. Witness reputation is based on other agents’ observations about the agent being evaluated. Each agent maintains a list of known agents that can provide it with reputation information [HUY04]. In determining witness reputation, the evaluating agent sends a query to the known agents [HUY04]. Upon receiving the request the known agents will search for ratings in their own local ratings database [HUY04]. If relevant ratings are found they are returned to the evaluating agent. If no matching ratings are found the known agent will instead refer the evaluating agent to other agents that it believes will be of assistance. The returned ratings are combined in the same manner as the ratings for direct interaction to produce a value for witness reputation [HUY04]. The reliability value for witness
reputation is also calculated in the same manner as the reliability value for direct interaction [HUY04].

The final aspect of the FIRE trust model discussed here is the use of certified reputation. Certified reputation is analogous to providing references when applying for a job [HUY04, HUY06]. After every interaction an agent may ask its interaction partner to provide a rating of its performance [HUY04, HUY06]. The rating is stored for future use. At a later point in time the agent may be requested to provide references of its past performance prior to interaction with a new partner [HUY04, HUY06]. The agent will supply its best references to the potential interaction partner. The potential interaction partner will use the references in evaluating trustworthiness [HUY04, HUY06]. The value for certified reputation along with its associated reliability value is calculated as per witness reputation [HUY04, HUY06].

The resulting values of interaction trust, role-based trust, witness reputation, and certified reputation are combined to produce a single measure of trustworthiness [HUY04]. The combination is achieved through the calculation of a weighted average. Similarly, a single associated reliability value is also computed. The weights can be set by the user to ensure the influence of each component [HUY04].

The preceding paragraphs and section 4.6 as a whole summarise the FIRE trust model. Similarly, section 4.5 summarises REGRET; section 4.4 summarises TRAVOS; section 4.3 summarises the information trust model; and section 4.2 summarises the FCM model. The sections that follow critically analyse the surveyed models.

**4.7 Critical Evaluation**

**4.7.1 Criteria for Evaluation**

In order to evaluate the selected methods of trust evaluation it is necessary to specify the criteria upon which the evaluation will be based. To determine such criteria is to ask what makes a method or model suitable for the purposes of evaluating trust in an agent or an object.
The criteria to be specified can be categorised as either functional criteria or non-functional criteria. A functional criterion specifies the features the model should exhibit. Such a criterion does not specify the manner in which functionality must be achieved. On the other hand, a non-functional criterion relates to the manner in which functionality is to be achieved. The following paragraphs specify and motivate both functional and non-functional criteria.

4.7.2 Functional Criteria
The functional criteria presented below were derived by consideration of the design goals of the models presented in the literature [CAS03, HUY04, SAB01, SIE05, TEA06]. For a model or method to be suitable for the purpose of evaluating trust it must:

- Allow an agent to model its trust in another agent or object.
- Allow for the implementation of different personality heuristics.
- Provide a trust metric to allow for comparisons.
- Allow for the proportional aggregation of factors, taken into consideration in the calculation of trust, relative to the perceived confidence and importance of those factors.

Obviously the model or method in question should allow for an agent to model its trust in other agents and objects. The modelling of trust is the fundamental purpose of such a model or method and if it fails in this regard then it fails completely. It is important to note that an agent should be able to model its trust in objects. An agent will invariably encounter objects, services and other tools in its environment. It will be helpful if an agent can reason about their reliability.

Castelfranchi et al motivate the relevance of the second criterion [CAS03]. If it is possible to model personality traits such as optimism or pessimism then an agent is able to reach a decision in cases where indecision may result. Such a case could occur when an agent has little direct experience (and knowledge of an agent) and the acquired reputation information diverges (i.e. inconsistencies exist amongst reputation reports).
The last two criteria were mostly derived from the design goals of TRAVOS and FIRE [TEA06, HUY04]. Typically an agent will evaluate the trustworthiness of several agents as part of deciding which agent to delegate a task to. Thus it is important to be able to have a measure of trustworthiness that allows for comparisons.

Trust evaluation is usually based on numerous factors. It is naïve to consider all factors equal in weight. Furthermore, the agent concerned may have knowledge that leads to greater confidence in some factors. Thus it is necessary for a model or method to allow for the dynamic scaling of the weighting of each factor considered in computing trustworthiness.

### 4.7.3 Non-Functional Criteria

In addition to, and just as important as, the above mentioned functional criteria are the following non-functional criteria derived from the literature [FUL04, HUY04]:

- Robustness.
- Reliability.
- Adaptability.
- Efficiency.
- Reusability (including generality, flexibility and scalability).

For a method or model to be suited to evaluating trust it must be robust in that it must be resistant to deceptive agents [HUY04]. If a deceptive agent can easily and frequently deceive another agent despite the method or model employed, then clearly the model or method is ineffective. Furthermore in extreme cases, such a model is detrimental because time and resources are wasted by the evaluating agent [CAS98]. A method or model should also be robust in the sense that it should still function when some sources of trust are unavailable [HUY04]. In addition to the robustness criterion Fullam and Barber assert that trust models must be reliable, adaptable, efficient, scalable, generic, and flexible [FUL04]. The non-functional criteria are subsequently discussed.

The model or method for evaluating trust should be reliable. To achieve reliability the model must provide an accurate estimation of another agent’s trustworthiness or
reliability [FUL04]. As mentioned above, if the model or method is not reliable the model or method in question is detrimental for the evaluating agent.

Any model or method should be adaptable in the sense that it should remain functional in a changing environment [FUL04]. Adaptability is particularly important in open mobile agent systems where agents may leave their immediate environment. Similarly new agents may arrive in such an environment.

A model or method should also be efficient for agents to use. If evaluating another agent’s trustworthiness takes too long or requires too many resources then the evaluating agent may be placed at a disadvantage. Of particular importance is the time taken to arrive at consistent and accurate estimates of a previously unknown agent’s trustworthiness [FUL04]. The performance related criterion is perhaps more important in dynamic environments.

Perhaps the most important non-functional criterion is that of reusability. Ideally speaking a model or method should be applicable to a range of situations, domains etc. Of particular interest is the reusability of a model or method. The method or model in question is likely to be reusable if the following criteria all hold [FUL04]:

- Generic – meaning that the model is applicable to numerous distinctly different domains.
- Scalable – the model can be applied in both smaller systems but is also suited to large multi-agent systems.
- Flexible – the model must not only allow the evaluating agent to evaluate the trustworthiness of another agent, but also the reliability of an object, service, or tool.

To motivate the importance of reusability consider that flexibility is quite pertinent in mobile agent systems where an agent may need to evaluate the trustworthiness of a host server. If a model does not exhibit such flexibility then it is clearly not reusable in the mobile agent system.

The evaluation of the selected methods and models in chapter 4 does not categorise a model or method as good or bad. It is acknowledged that the efficacy of any model is
relative to the situation and domain in which it is used. Instead, the evaluation looks at the models and methods from a holistic perspective.

### 4.7.4 Evaluating the FCM Model

The use of FCMs in trust evaluation is a simple, yet effective method for evaluating trust. The FCM approach satisfies most of the criteria laid out above. The paragraphs that follow evaluate the approach based on the specified criteria.

The use of FCMs clearly allows an agent to model its trust in another entity [CAS03, FAL04b]. Furthermore, the use of FCM allows for the specification of different personality heuristics [CAS03]. The designer of the FCM can specify a pessimistic, risk averse agent by manually setting edge weights appropriately. By specifying personality factors one particular belief such as a harmfulness belief can have a greater influence on the overall trust evaluation.

The FCM approach to evaluating trust does result in a numerical value which is ‘squashed’ into a fuzzy interval using a threshold function [CAS03]. Numerical trust values could be used for comparison. Alternatively, the comparison could occur at the fuzzy interval level. The advantage of the fuzzy approach is that minor and negligible differences in trustworthiness are ignored [CAS03].

Perhaps the greatest strength of the FCM approach lies in its robustness. Firstly, several sources of trust are taken into account meaning that even in the absence of one or other source of trust it is still possible to evaluate another agent’s trustworthiness. There is also evidence to suggest that the cognitive approach is robust in the presence of deceitful agents [FAL04b]. Falcone, Pezullo, Castelfranchi, and Calvi present the results of experiments where even after ‘noise’ in the input data is introduced the implementation performs well [FAL04b]. The experiment shows that even if some inaccurate values for beliefs or belief sources are specified, the implementation still performs reasonably well. The above mentioned experiments also indicate the relative reliability and efficiency of the method in question. Intuitively however, efficiency is inversely proportional to the size of the FCM. Thus the FCM approach could be computationally expensive.
It can also be argued that the FCM model for trust evaluation is generic and thus reusable. Clearly, FCM can be applied in different domains and thus it is generic [CAS03]. Furthermore, it can be argued that the method could be applied both to evaluating the trustworthiness of another cognitive agent or a simple service. Therefore the method is flexible. However, the scalability of the method is questionable; for every trust evaluation of a distinct entity, a different FCM is required [CAS03]. Therefore if the evaluating agent needs to evaluate hundreds of agents, objects, or services then the time and space requirements could become problematic. Furthermore, difficulties may be encountered in environments with large numbers of agents. Difficulties may arise because the evaluating agent may frequently encounter new agents but lack the information it needs to produce a reliable result [HUY04]. In conclusion, the method is potentially reusable in most cases.

Perhaps the biggest criticism stems from the need for human involvement. The design of the FCM including the specification of edge weights is done manually [CAS03]. The need for human intervention in the trust evaluation process may undesirably affect the autonomy of the agent.

The need for human intervention in the assignment of the edge weights may also translate into a ‘one size fits all’ FCM which is potentially problematic. The adaptability of the model in dynamic environments (open environments in particular) could be hampered unless the FCM can be adjusted dynamically. Furthermore, while the method may allow for the evaluation of entities other than cognitive agents – it does so inefficiently. Determining the reliability of a service or object does not require as complex an FCM as is needed when evaluating the trustworthiness of a cognitive agent (See chapter 3).

Ideally, an agent should be endowed with knowledge of a user’s personality or preferences. Using the knowledge the agent should be able to dynamically create an appropriate (not unnecessarily complicated) FCM for each entity that it encounters. Dynamic FCM creation would ensure autonomy and improve the degree of adaptability as well as performance. The use of knowledge in FCM creation also opens up the possibility for an agent to learn the best design for an FCM in a range of circumstances [CAS03].
4.7.5 Evaluating the Information Trust Model

The information trust model allows an agent to evaluate another agent’s trustworthiness and thus partially satisfies the first functional criterion of allowing an agent to model its trust in another agent or object. Trust is modelled using a measure of uncertainty so one agent’s trust in another is at best modelled as a number [SIE05]. Therefore the reasons for the value of trust are not inherent in the information trust model.

The second functional criterion dealing with the implementation of different personality heuristics is implicitly satisfied. The information trust model could potentially cater for personality heuristics through beliefs in its belief base [SIE05]. However, there is no explicit means for implementing personality heuristics. As far as comparison is concerned, it is assumed that some comparison is possible since trust is represented by a number. The assumption is supported by the fact that trust plays a role in the agent’s negotiation strategy [SIE05]; should an agent negotiate a deal with multiple agents then the agent concerned must be able to compare the trustworthiness of its potential partners.

In considering the non-functional criteria, several inadequacies become apparent. The robustness of the model is questionable. In the absence of social information and knowledge of other agent’s preferences the evaluating agent can only rely on recent direct experience [SIE05]. Furthermore, it is quite possible that in environments with large numbers of agents the evaluating agent will have access to neither the relevant social information nor information relating to other agents’ preferences, meaning that trust evaluations and agent negotiation will be negatively affected. Assuming the agent has access to the relevant information, there is little or no evidence to suggest that the model lacks robustness in the context of deceitful agents [SIE05].

Similarly, there is little or no evidence in the literature implicating the model as unreliable [SIE05]. However the model could potentially result in an unreliable estimation of an agent’s trustworthiness since it does not take into account external, environmental factors [CAS98, CAS01b, T3G].
Another disadvantage of the information trust model is its complexity. From a practical point of view it would require knowledge of mathematical statistics in order to implement it. Furthermore, the computation of trust values using the information trust model is computationally expensive (due to the extensive floating point operations required) although not prohibitively so.

In terms of adaptability, there is once again little or no evidence in the literature to suggest that an agent implementation of the information trust model would not be able to adapt to changes in the environment [SIE05]. However, if the environment became somewhat more hostile then the information trust model may be less reliable thus calling into question the degree of adaptability.

Perhaps the biggest criticism is the generality of the information trust model. Admittedly, the model was designed specifically for environments where agents negotiate implying that the model is not particularly generic because it is only well suited to domains where agent negotiation is required [SIE05, DEB06]. Furthermore, the model was not designed for anything other than negotiating agents and thus is not flexible [SIE05]. In terms of scalability, the computational complexity of the information trust model could be questioned. More problematically however, would be the lack of information associated with a substantial increase in the numbers of agents. Therefore it is clear that the reusability is limited.

4.7.6 Evaluating TRAVOS

TRAVOS models the degree of trust in an agent as the expected value of the unknown probability of a successful outcome [TEA06]. As a consequence, the degree of trust is represented by a number making comparison possible. Furthermore, the specification of a confidence metric means that the perceived confidence of an evaluation is taken into account [TEA06].

The TRAVOS model seems to satisfy most of the functional criteria except one. The model does not seem to allow for the specification of personality heuristics. At best it is possible to model a more cautious or risk-averse agent by requiring a higher confidence threshold.
TRAVOS uses at most two sources of trust [TEA06]. Therefore if an agent has little direct experience and there is little or no reputation information available unreliable trust evaluations will be derived. It can be concluded that in the face of unavailable sources of trust, TRAVOS cannot be considered particularly robust. The empirical evaluation of TRAVOS presented in the literature suggests that TRAVOS is rather robust in terms of dealing with misleading agents [TEA06]. However, the TRAVOS agent is vulnerable for a short period of time if it does not have enough direct experience (with both the agent being evaluated and the reputation sources) and the reputation sources are misleading.

The above cited empirical evaluation also confirms the reliability of TRAVOS [TEA06]. The results of the experiments also confirm that the reliability of the model increases with direct interactions meaning that the time taken for the model to converge to an accurate trust value is dependent on the rate of direct interactions. Therefore the rate of direct interactions affects the efficiency of the model; if the rate of direct interactions is low then timely convergence to accurate estimates of trustworthiness may be hampered. From a more practical perspective, the model is computationally expensive (although not prohibitively so unless dealing with large amounts of reputation information) requiring the calculation of several probabilities and integrals [ZHA06].

According to the literature the current model does not consider the possibility that the behaviour of agents may evolve or change over time [TEA06, ZHA06]. Therefore, currently TRAVOS is not particularly adaptable. TRAVOS also does not distinguish between environmental factors and internal, agent factors [TEA06]. The lack of attribution could also inhibit its adaptability.

TRAVOS is a fairly general model and it seems that there is little or no reason prohibiting its application in varying domains [TEA06]. Furthermore, TRAVOS could be used for evaluating the reliability of an object, service, or tool. Thus the model is flexible. Given the perceived generality, scalability, and flexibility of TRAVOS, it appears to be reusable. The only question that remains is that of scalability. In really large MASs with many agents, the TRAVOS agent may not have
access to useful reputation providers and may lack the experience. Thus it can be said that the TRAVOS model is suited to reuse in most cases.

4.7.7 Evaluating REGRET

In the REGRET system trust is modelled as a function of the individual, social, and ontological dimensions of reputation [SAB01, MUI02]. Ultimately, trust is represented by a number allowing for comparisons. The various metrics forming the social dimension of reputation allow for an even more detailed comparison.

As far as REGRET is concerned the model itself does not provide for the implementation of different personality heuristics. Although REGRET does not provide for the implementation of personality heuristics, it is possible that personality factors could be taken into account in the forming of the agent’s subjective opinion that is the basis of individual reputation (the equivalent of evaluating direct interaction) [SAB01].

REGRET also takes into account confidence values for its evaluations [SAB01]. Furthermore, the model allows the various components of the total trust to be additively combined using a weighted sum [SAB01]. The additive combination of the components of the total trust allows for a so called proportional aggregation of factors.

In terms of robustness the REGRET model takes into account multiple sources of trust [SAB01, MUI02, BIL07]. The use of the individual dimension of reputation is equivalent to direct experience. The social dimension takes into account factors such as group reputation and opinions of those agents in the relevant social groups. The ontological dimension exploits categorisation as a source of trust. Thus the model is relatively robust. However, new agents with little direct experience, who know few other agents, will struggle to evaluate trustworthiness accurately.

In terms of robustness in the context of misleading agents, the model is potentially fallible if there are many deceitful agents and the evaluating agent does not suspect their deceit. The fallibility is due to REGRET’s reliance on social reputation
However, in most simple cases the model performs well – as shown by the results of experiments presented by Sabater and Sierra [SAB01].

In terms of efficiency the model should converge quickly to an accurate level of trust if accurate reputation information is available. The time taken for the model to converge to an accurate value of trust is also partly dependent on the rate of direct interaction [SAB01].

REGRET appears to be a reliable trust model. The results of the experiments presented by Sabater and Sierra [SAB01] seem to corroborate that the REGRET model produces agent behaviour as good as some other systems compared in their study. The results also seem to indicate that the use of social reputation measures improved behaviour [SAB01]. Thus the results of their experiments attest to the reliability of REGRET.

As far as adaptability is concerned there is little or no evidence in the literature to suggest that REGRET fails to function in the event of changes in its environment [SAB01]. However, once again in environments with many agents the REGRET agent may find itself lacking the necessary reputation information and experience from time to time.

The REGRET model is generic. The discussion of the experiments from the previously cited literature mentions the application of the model in two different domains [SAB01]. The use of REGRET in different domains supports claims of its generality. There is also nothing preventing the model being used to evaluate the reliability of objects, services or tools. Thus REGRET is flexible. In terms of scalability, the model scales well so long as the implementing agent always has access to reliable reputation information. However, access to reliable reputation information can seldom be guaranteed in large, open MAS with many unknown agents [HUY04]. Given the above discussion it is reasonable to conclude that the REGRET model will lend itself to reuse most of the time.
4.7.8 Evaluating FIRE

The FIRE model allows for the decentralised evaluation of agents [HUY04]. Trust is represented numerically and therefore comparisons between trust evaluations are certainly possible.

The model clearly allows for the aggregation of factors (interaction trust, role-based trust, witness reputation, and certified reputation) and takes into account the perceived confidence in the respective factors [HUY04, HUY06]. The use of different sources of trust is a key strength of FIRE. In large open environments new agents in particular may struggle to reach trust evaluations due to a lack of information [ASH05, HUY04, TEA06]. The use of certified reputation information is in theory useful because it is likely to be readily available [HUY04, HUY06]. However, the reliability of certified reputation is questionable. The agent concerned will only forward its best references and it is foreseeable that in some cases those preferred references may be overly high. Admittedly, the certified reputation information can be adjusted to counterbalance any over estimation [HUY04, HUY06]. Nonetheless, the use of certified reputation may lead to trust evaluations where trust evaluations would otherwise have been impossible due to a lack of information.

The FIRE model does not explicitly take the modelling and implementation of different personality heuristics into account. However, the user is able to specify weights for the aggregation of factors and in so doing personality factors (aversion to risk for example) may implicitly influence the evaluation of trustworthiness.

The FIRE model is likely to exhibit a degree of robustness due to the use of multiple sources of trust. However, the model assumes that agents provide honest reputation information [HUY04]. Furthermore, there is no real incentive for the agents to provide honest reputation information. Therefore, the robustness of the model is hampered. It is however suggested that future work on FIRE will address the robustness concern [HUY04].

The model is thought to be reliable in the sense that even in situations where an agent lacks information (i.e. when it joins the environment as a new agent) the agent is able to reach an evaluation. Furthermore, it is claimed that witness reputation leads to
more accurate evaluations [HUY04]. It is true that the use of different sources of trust may lead to improved evaluations on the basis of the assumption that all the different sources of trust are reliable.

The FIRE model exhibits a degree of adaptability. Due to the use of different sources of trust, including certified reputation, FIRE remains functional in a changing environment [HUY04]. However, FIRE also requires user involvement in the setting of certain parameters (the influence of each component for example) [HUY04]. The manual setting of parameters limits the adaptability of FIRE to changing environments. The dynamic adjustment of certain parameters to enhance adaptability is an area for future work [HUY04].

The efficiency of FIRE is debatable. The main question of the FIRE model’s efficiency stems from the use of all four components (interaction trust, role-based trust, witness reputation, and certified reputation) all of the time. The use of witness reputation and certified reputation involves agent communication and can be time consuming [HUY04, HUY06]. Furthermore, the criticism of the decentralised approach to trust evaluation is that each agent is then responsible for the potentially time consuming trust evaluation task. A potential improvement may be to use the various components only as necessary, in a similar manner to the TRAVOS model [TEA06].

The modular design and parameterised configuration of FIRE makes it generic [HUY04]. The model is therefore suited to use in different domains [HUY04]. FIRE exhibits a degree of flexibility, so long as objects, services, or tools can be modelled as agents. Obviously, the use of certified reputation cannot readily be employed if evaluating the trustworthiness of an object, service, or tool as opposed to an agent. The FIRE model is also expected to be scalable. Although FIRE is clearly designed for large open multi-agent systems [HUY04], it is foreseeable that it could be deployed in various environments, including smaller closed systems. Therefore, it can be concluded that the FIRE model can be reused in a range of different situations.
4.7.9 Comparison of Evaluations
Section 4.7.9 briefly summarises the evaluation of the models presented in chapter 4. The comparison provided in way of summary compares the models presented based only on the non-functional criteria. The comparison is presented in tabular format in table 4.1. The criterion relating to a particular model is simply classified as being poor, average or good.

4.8 Conclusion
Chapter 4 builds on chapter 3 by looking at existing approaches to trust evaluation. Several models are considered. Amongst the models considered is the FCM approach. The FCM approach is a cognitive approach that implements the decomposition of trust into its cognitive constituents. The TRAVOS model uses probability theory to model one agent’s trust in another. The model attempts to handle reputation information and also demonstrates the use of confidence or credibility values. REGRET is a reputation-centric approach. The FIRE model relies on several sources of trust including certified reputation. In contrast to the FCM, TRAVOS, and REGRET, and FIRE models, the information trust model attempts to avoid uncertain estimates and instead focuses on what is known. The information trust model also introduces the idea of information decay.

Perhaps the most salient contribution of the chapter is the criteria for evaluating trust evaluation approaches. The criteria are discussed, motivated and subsequently applied to the models discussed in chapter 4.

The penultimate section attempts to compare various models aimed at evaluating trustworthiness. According to the tabular comparison, the strengths of the FCM approach are apparent. The FIRE model, as a more recent contribution, is also thought to be a superior model. Despite the analysis and comparison presented in chapter 4 it should not be forgotten that all the models presented here were quite possibly subject to different design criteria and the designers themselves may have had different ends or application domains in mind. As such evaluation and comparison is difficult and all the models have their value.
<table>
<thead>
<tr>
<th></th>
<th>Robust: Trust Sources</th>
<th>Robust: Misleading Sources</th>
<th>Reliability</th>
<th>Adaptability</th>
<th>Efficiency: Convergence</th>
<th>Efficiency: Computational Complexity</th>
<th>Flexibility</th>
<th>Scalability</th>
<th>Generic</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCM</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Average</td>
<td>Good</td>
<td>Average</td>
<td>Good</td>
</tr>
<tr>
<td>Information Trust Model</td>
<td>Poor</td>
<td>Good</td>
<td>Good</td>
<td>Poor</td>
<td>Average</td>
<td>Poor</td>
<td>Poor</td>
<td>Average</td>
<td>Poor</td>
</tr>
<tr>
<td>TRAVOS</td>
<td>Poor</td>
<td>Good</td>
<td>Good</td>
<td>Poor</td>
<td>Average</td>
<td>Poor</td>
<td>Good</td>
<td>Average</td>
<td>Good</td>
</tr>
<tr>
<td>REGRET</td>
<td>Average</td>
<td>Good</td>
<td>Good</td>
<td>Average</td>
<td>Average</td>
<td>Good</td>
<td>Good</td>
<td>Average</td>
<td>Good</td>
</tr>
<tr>
<td>FIRE</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Average</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
</tr>
</tbody>
</table>

Table 4.1 A Tabular Comparison of Trust Models
The work presented in chapter 4 is indicative of the state of the art as far as trust and reputation models are concerned. The chapter contributes to the dissertation by providing a clear indication of existing approaches to trust evaluation. Chapter 4 has built on chapter 3 by revealing how an agent may evaluate trustworthiness. Furthermore, chapter 4 contributes by outlining criteria that a model should meet to be deemed suitable for the purposes of trust evaluation. The chapter that follows furthers the investigation into the state of the art by considering some important ancillary contributions.
Chapter 5
Trust and Reputation: The State of the Art

5.1 Introduction
Chapter 5 presents supplementary methods and techniques aimed at improving the evaluation of trustworthiness. In so doing, chapter 5 builds on chapter 4 by providing a more thorough investigation into the state of the art as far as trust and reputation are concerned.

It is expected that the fundamental concepts of the methods and techniques presented in chapter 5 can be employed in the future design and development of a model for evaluating trustworthiness. It is further expected that the methods and techniques that are the topic of chapter 5 could augment existing approaches to trust evaluation making them more robust. Enhancing existing approaches by incorporating the ancillary contributions outlined in chapter 5 should result in models and methods that better satisfy the criteria set out in chapter 4.

The chapter identifies contributions along two dimensions: the ontological dimension and the methodological dimension. Each contribution is discussed and analysed. The analysis motivates why the contribution is significant but also points out any identified shortcomings.

The following section starts by considering contributions at the ontological level.

5.2 Ontologies for Trust and Reputation
5.2.1 Introduction
Section 5.2 presents and analyses that which is perceived to be a significant ontological contribution to the discipline. Casare and Sichman presents a ‘functional ontology of reputation’, the purpose of which is to codify knowledge of reputation and represent it in a machine readable and structured form [CAS05]. The attempt to design and implement an ontology of reputation is described in the following
5.2.2 A Functional Ontology of Reputation

Casare and Sichman acknowledge that there is a considerable base of reputation related knowledge that can be applied to virtual, electronic environments [CAS05]. The origins of reputation related knowledge is in disciplines such as sociology, psychology and even economics [CAS05]. Casare and Sichman attempt to aggregate reputation related knowledge from the various disciplines and represent it in an ontology [CAS05].

Casare and Sichman adopt categories of knowledge analogous to those of the Functional Ontology of Law [CAS05]. The adoption of analogous categories is based on the claim that ‘the concepts of the legal world can be used to model the social world’. The four categories of knowledge adopted by Casare and Sichman form a framework that supports many intuitive concepts of reputation derived from sociological, psychological and economic works [CAS05]. Each of the four categories of knowledge is briefly discussed subsequently. How the four categories of knowledge relate in practice, also receives attention.

The category of knowledge that is most reputation oriented is the aptly termed ‘Reputative Knowledge’ category. The reputative knowledge category deals with the semantics of reputation; what it means for an agent to have a good reputation or a bad reputation and so forth [CAS05]. The reputative knowledge category also categorises and characterises reputation. For example a reputation nature concept is specified. Furthermore, different types of reputation are distinguishable; individual reputation, group reputation, product reputation are all specified [CAS05, VER07]. Additionally, reputation type concepts are also defined allowing reputation to be distinguished based on the information source from which it was derived. Figure 5.1 (adapted from [CAS05]) aims to illustrate the type of knowledge represented by the reputative knowledge category. However, it is also important to note that the reputative category of knowledge is quite comprehensive. Figure 5.1 below is by no means complete, and a full discussion of all the concepts is beyond the scope of the dissertation.
Responsibility knowledge is another category of knowledge. The responsibility knowledge category consists of concepts such as attribution notion, actor-observer effect, responsibility assignment and responsibility restriction [CAS05]. The purpose of responsibility knowledge is that it allows an agent to attribute a cause to agent behaviour and thus determine to what extent the agent is responsible. Therefore responsibility knowledge is particularly useful in the context of the cognitive trust theory of chapter 3. Cognitive trust theory promotes the attribution of a success or failure to either factors internal to the agent (within its region of influence) or to factors beyond its control (external or environmental factors) [FAL04a].

The normative knowledge category is intended to allow an agent to characterise and categorise another agent’s behaviour [CAS05]. Using normative knowledge, an agent should be able to classify agent behaviour as good or bad, adequate or inadequate. Therefore, the normative category of knowledge models the concepts of social normality. The development of the categories of normative knowledge and world knowledge is incomplete and constitutes future work [CAS05].
The world category of knowledge consists of knowledge about the world or environment in which the agents and their social customs exist [CAS05]. The purpose of the knowledge in the world category is to allow an agent’s social behaviour to be described. The description is an important input into an agent reasoning process that incorporates the semantics of the functional ontology of reputation as is subsequently discussed.

Consider figure 5.2 (adapted from [CAS05]). Figure 5.2 describes how an agent can use the knowledge represented in the Functional Ontology of Reputation in a reasoning process. The purpose of the reasoning process is to arrive at an accurate and machine understandable description of reputation. An agent’s actions are perceived. The agent’s actions that ultimately constitute agent social behaviour are interpreted using the world knowledge category of the ontology [CAS05]. The machine understandable description of behaviour is then used as input to a reasoning process where the agent can determine if the behaviour is adequate or inadequate. Determining if the behaviour is adequate or inadequate is achieved using the concepts that comprise the normative knowledge category [CAS05]. The same behaviour description is also used in the attribution of the behaviour. Specifying such an attribution requires concepts from the responsibility knowledge category [CAS05]. Finally, using the above knowledge as per figure 5.2, it is possible to articulate reputation in a concise and machine understandable way.

The reputation knowledge can then be conveyed to other agents. The reputation knowledge will ultimately be useful to them particularly given the shared semantics.

Sub-section 5.2.2 describes the Functional Ontology of Reputation. It also attempts to convey how the functional ontology of reputation could be used. The Functional Ontology of Reputation is useful for interpreting agent social behaviour and classifying it accordingly. Once an opinion of the agent (based on the social behaviour) is formed, the opinion can be shared with other agents as reputation information described in terms of the concepts of the ontology [CAS05, VER07]. The following sub-section elaborates on the relevance of the Functional Ontology of Reputation.
5.2.3 The Relevance of the Functional Ontology of Reputation

The aforementioned Functional Ontology of Reputation is important because it allows agent’s within a society to share reputation information. Without a trust and reputation ontology and thus without shared semantics it becomes near to impossible for agents to accurately share information [DAC03, CAS05, VER07]. One agent’s knowledge or understanding of reputation may differ from another’s and hence a common semantics is required in order to share reputation information. For example, consider the concept ‘is reliable’. What precisely does ‘is reliable’ mean? If two distinct agents do not share the same knowledge or understanding of the concept ‘is reliable’, then how can they accurately share knowledge of another agent’s reputation?

Other authors have also identified this problem and motivate the need for ‘the development of a semantics for trust to allow sharing of trust information’ [GRI05].
Griffiths motivates the need for a more comprehensive ontology to remove the subjectivity from loosely defined concepts.

Mui, Halberstadt and Mohtashemi also recognise the importance of the classification of reputation [MUI02]. However, they stop short of specifying a comprehensive ontology.

Clearly various researchers have seen the need for ontologies of trust and reputation. As a consequence, numerous ontologies could be developed. One may question how numerous ontologies could possibly help reach the goal of a shared semantics especially in open environments? Indeed the use of numerous ontologies by different entities results in the so-called ‘semantic mapping problem’. The semantic mapping problem is commonly encountered in the context of the Semantic Web [DAC03]. It is in solving the semantic mapping problem that the Functional Ontology of Reputation has another potential use. Casare and Sichman suggest using it as a meta-ontology for the purposes of semantic mapping [CAS05, VER07]. Using the Functional Ontology of Reputation as a meta-ontology will allow agents using different internal reputation models in open systems to share reputation information [VER07].

Ontologies for trust and reputation are useful for sharing reputation information unambiguously. However, the use of ontologies in general is not without problems (the semantic mapping problem for example). Sub-section 5.2.4 looks at potential future improvements aimed at improving the usability of the Functional Ontology of Reputation.

5.2.4 Improvement and Future Work

Casare and Sichman aim to develop the Functional Ontology of Reputation by fully specifying the normative and world knowledge categories [CAS05]. In so doing they will have aggregated reputation knowledge from various disciplines. Furthermore they will have arranged it in an appropriate structured form.

As already mentioned, other researcher’s (not necessarily even working in the same domain) will continue to develop their own ontologies. Clearly it is important to have a working semantic mapping process. However, in practice it is quite possible that a
semantic mapping process will be inadequate. The reality is that in really large open systems such as the Semantic Web, the issue of pragmatics (context of use, intent of semantics) cannot be ignored. Issues such as relevance of information in the context of information overload become problematic [deM05]. Thus it is perhaps necessary to move beyond semantic mapping (meaning alignment) towards meaning negotiation at the pragmatic level. Such a strategy is outlined by de Moor [deM05].

The following paragraph summarises the work presented so far.

5.2.5 Summary
Section 5.2 presents the use of ontology in trust evaluation. The particular contribution considered is the Functional Ontology of Reputation, which has an analogue in the Functional Ontology of Law [CAS05]. The Functional Ontology of Reputation aggregates reputation knowledge from other disciplines and structures it in an appropriate form. The ontology has four categories of knowledge which can be used in a reasoning process to allow an agent to characterise and classify another agent’s reputation [CAS05]. The Functional Ontology of Reputation allows for reputation information to be shared among agents that share the Functional Ontology of Reputation. The Functional Ontology of Reputation can also be used to facilitate semantic mapping so that agents with differing semantic models can share reputation information [CAS05, VER07].

5.3 Social Network Analysis & Relationship Analysis
5.3.1 Introduction
Section 5.2 looked at a significant contribution at the ontological level. Section 5.3 looks at one of several contributions at the methodological level. The paragraphs that follow focus on relationship analysis and social network analysis for the purposes of evaluating an agent’s trustworthiness. Section 5.3.2 starts by looking at the extraction of reputation information through social network analysis. The usefulness of social network analysis is motivated in section 5.3.3 and a summary is provided in section 5.3.4. Section 5.3.5 focuses on trust evaluation through relationship analysis. Section 5.3.6 motivates the importance of the contribution.
5.3.2 Social Network Analysis
The use of reputation in trust evaluation typically involves the participation of reputation providing agents. An alternative mechanism for reputation information elicitation based on social network analysis is proposed in the literature [ESF01, PUJ02, SAB02].

According to Pujol, Sanguesa and Delgado, a social network is ‘a representation of the relationships existing within a community’. A social network typically takes the form of a graph where the nodes represent the members of the community (agents) and the edges represent relationships between them [ESF01, PUJ02, SAB02]. Thus it is possible for multiple social networks to exist for a single community, each differing according to the type of relationship an edge represents.

The theory behind social network analysis is that the position of an agent in a graph can be used to infer certain reputation information [PUJ02]. For example, an agent that is revered, perhaps due to superior competence will be represented by a highly connected node.

This poses the question of how to build a graph where highly reputable agents are represented by highly connected nodes? Pujol et al focus on an application of social network analysis in a community of practice in the context of knowledge management [PUJ02]. Thus their approach to building the social network is not particularly generic. However, it seems intuitive that if an agent could acquire basic information about agent interactions it becomes possible to build the network. Such a network would have to be continually updated due to its dynamic nature.

Sub-section 5.3.2 describes social network analysis. The following sub-section elaborates on the prior presentation of social network analysis by motivating its relevance.

5.3.3 The Relevance of Social Network Analysis
As already mentioned, the use of reputation information in trust evaluations typically requires the involvement of a reputation provider. For a rational agent there is usually little incentive to provide reputation information. The particular issue of a lack of
incentive to provide reputation information is addressed in more detail in section 5.4. Ultimately for an agent to communicate reputation information or indeed, any kind of feedback (that could be used to establish reputation information) is time consuming. Therefore using reputation in trust evaluation is not without its problems.

The elicitation of reputation information via social network analysis as discussed in the previous sub-section solves some of the difficulties associated with acquiring reputation information from other agents [PUJ02, SAB02]. The application of social network analysis is promising and is clearly useful in the knowledge management ‘communities of practice’ domain. However, social network analysis is not without its problems. It exchanges the problem of acquiring reputation information from agents, for the problem of acquiring relationship information.

Clearly social network analysis is relevant, although there are issues still to be resolved. The following paragraph briefly summarises the section.

5.3.4 Summary
Section 5.3 discussed the use of social network analysis to infer reputation information. In a social network, an agent is represented by a node whilst the edges in the graph represent a certain type of relationship between agents. An agent that is highly connected is highly regarded in the society. Social network analysis avoids the problems of acquiring reputation information from other agents, but introduces the problem of acquiring the necessary information to build the social network graph.

The following section discusses relationship analysis – a similar and related contribution.

5.3.5 Relationship Analysis
Relationship analysis as introduced by Ashri et al [ASH05] is a generic method aimed at augmenting existing approaches to trust evaluation. Relationship analysis consists of the processes of relationship identification, relationship characterisation and relationship interpretation [ASH05]. Each process is subsequently discussed.
The relationship identification process allows an agent to identify the existence of relationships among agents. Each agent has a region or sphere of influence and a viewable environment [JEN01]. If an agent can acquire knowledge of how the relevant agents’ viewable environments and spheres of influence overlap then it becomes possible to identify the existence of relationships. Knowledge of an agent’s goals is also helpful in the relationship identification process [ASH05]. In practice, the use of ontology for the purpose of relationship identification could be useful.

After the relationships have been identified, they need to be characterised and classified. In order to characterise and classify relationships, the developers of the model have specified several types of relationship. The most atomic types of relationship that have been defined include trade, dependency, competition, and collaboration relationships [ASH05]. According to Ashri et al each relationship type has a corresponding configuration of viewable environments and spheres of influence [ASH05]. The configuration or pattern of viewable environments and spheres of influence allows for the actual relationship to be identified, and thus characterised and classified.

Relationship interpretation is a process whereby the evaluating agent tries to infer another agent’s trustworthiness based on the characterised and classified relationship(s) identified between it and the other agent [ASH05]. The process of relationship interpretation is largely dependant on the specific trust model or method of trust evaluation used; trust is evaluated according to direct interaction and reputation almost as would normally be the case. However, any confidence values or weights for reputation information are adjusted according to the relationship information [ASH05].

Sub-section 5.3.5 describes how Ashri et al suggest augmenting a more traditional approach to trust evaluation with relationship analysis. The following sub-section motivates the relevance of relationship analysis.

5.3.6 The Relevance of Relationship Analysis
The more traditional approaches to trust evaluation are typically inadequate. An agent that is new to an environment may lack direct experience, and is therefore at risk.
Furthermore, such an agent may then rely on reputation information acquired from third parties. However, as Ashri et al rightly point out; reliance on reputation information simply transfers the problem of trusting an agent to the problem of trusting the reputation provider [ASH05].

Therefore a method is required to help overcome the problem of minimal experience and subsequent reliance on potentially misleading reputation information, at least to the extent where an agent that has recently joined a multi-agent system has a fighting chance. The use of relationship analysis goes some way to addressing the problem of minimal experience and subsequent reliance on potentially misleading reputation information. Furthermore, it is also fairly general in nature and can be adapted to various environments – not just electronic marketplaces [ASH05].

However, relationship analysis is not without flaw. It could be problematic for a new agent to quickly acquire knowledge of another agent’s sphere of influence and viewable environment. Information relating to spheres of influence and viewable environments is by no means guaranteed to be readily available. Therefore, relationship analysis is more suited to cooperative environments where information about agent capabilities is readily available in a public directory.

5.3.7 Summary
Section 5.3.5 summarises and discusses relationship analysis. Relationship analysis is a process that can be used to augment the more traditional approaches to trust evaluation. The process of relationship analysis includes identifying the existence of relationships between agents [ASH05]. It also comprises a process of characterising and classifying relationships. The relationships are interpreted and the credibility of reputation information is adjusted [ASH05]. Relationship analysis overcomes the shortcomings of traditional approaches – as was discussed in section 5.3.6.

5.4 Incentive Compatible Reputation Mechanisms
5.4.1 Introduction
The previous two sections focused on supplementary techniques for trust evaluation. Section 5.4 diverges from the topic of supplementary techniques for trust evaluation.
Section 5.4 instead focuses on a contribution to the state of the art that addresses a problematic issue associated with many reputation mechanisms.

The problematic issue at hand is how to achieve truthful reporting of reputation information. A solution that strives to ensure the honest reporting of reputation information is presented in sub-section 5.4.2. The importance of the solution is highlighted in sub-section 5.4.3.

5.4.2 An Incentive Compatible Reputation Mechanism

Jurca and Faltings’ incentive compatible reputation mechanism aims to elicit truthful reporting of reputation information from rational self-interested agents [JUR03, JUR06]. Such agents may otherwise be less than honest. Truthful reporting of reputation information is achieved through the introduction of special broker agents [JUR03]. The purpose of a broker agent is to buy and sell reputation information. Consider an agent X that wants to evaluate the trustworthiness of agent Y. X may choose to purchase reputation information about Y from a broker agent B. Since the broker agents’ reputation information is not necessarily synchronised, X could purchase information from more than one broker agent. After X has interacted with Y, X can then form an opinion about Y and sell it as reputation information back to the broker agent B and any other broker agents consulted [JUR03].

Broker agents’ sole purpose is to trade in reputation information. It is suggested that broker agents are designed with the goal of breaking even over a long period of time [JUR03]. The goal of breaking even is achieved by scaling the buying and selling of reputation information appropriately. The incentive mechanism as described up to this point provides an incentive for an agent to report reputation information truthfully; the agent receives compensation for doing so. However, an agent could still report false reputation information. The reporting of false reputation information is addressed through the implementation of a payment function.

The purpose of the payment function is to elicit honest reputation reporting [JUR03]. Ideally speaking, if an agent reports false reputation information that agent should not receive any compensation. Conversely, the payment function should be maximised in the event of honest reputation reporting. One example payment function presented in
the cited literature suggests that the broker agent only pays for reputation information if it is consistent with the relevant reports provided by other agents [JUR03].

Sub-section 5.4.2 briefly outlines an incentive compatible reputation mechanism as presented by Jurca et al [JUR03]. The incentive compatible reputation mechanism is not perfect; the success of it depends on a robust payment function which could be difficult to implement. Nevertheless, the incentive compatible reputation mechanism improves trust evaluation based on third party reputation information. The importance of the incentive compatible reputation mechanism is motivated next.

5.4.3 The Relevance of an Incentive Compatible Reputation Mechanism

The incentive compatible reputation mechanism discussed above encourages rational and naturally self-interested agents to provide reputation information. If there was no incentive for doing so the agents in question may simply decide against supplying reputation information. Agents may decide against supplying reputation information primarily for three reasons [JUR03]:

- The cost associated with the action of providing reputation information in terms of time and resources.
- By providing reputation information, other agents gain a competitive advantage. Thus in many environments it would not be rational to provide reputation information. Hence, there needs to be an incentive to do so.
- Should any reputable agent supply a positive reputation report relating to another distinct agent, the difference between the supplier agent’s reputation and the other agent’s reputation is reduced. Therefore, the value of the supplier agent’s reputation decreases relative to the other agent’s reputation.

Clearly an incentive compatible reputation mechanism is required. None of the trust models presented in chapter 4 appear to provide an incentive for agents to report reputation information. The incentive compatible reputation mechanism could possibly be used in conjunction with existing trust evaluation models.

The difficulty of such an approach is in determining an appropriate payment function that results in an agent receiving compensation for only honest reputation information. The following paragraph summarises the incentive compatible reputation mechanism.
5.4.4 Summary
The incentive compatible reputation mechanism aims to elicit truthful reporting of reputation information. Truthful reporting of reputation information is ultimately achieved by two key ideas. The first idea involves the introduction of broker agents to buy and sell reputation information. As a consequence agents sell reputation information to achieve compensation. The second idea involves the use of an appropriate payment function. The purpose of the payment function is to elicit honest reputation reporting. The following section concludes chapter 5.

5.5 Conclusion
Chapter 5 has presented significant contributions to the state of the art. Several supplementary techniques aimed at improving trust evaluation were presented. Furthermore, an explanation motivating the relevance of each technique has been supplied. In concluding chapter 5, a brief summary is presented. Further, an explanation relating to the importance of chapter 5 is provided.

Significant contributions have been identified at both the ontological level and the methodological level. Various contributions relating to the development of trust and reputation ontologies were highlighted. The most significant contribution is the Functional Ontology of Reputation. The Functional Ontology of Reputation aggregates and structures existing reputation related knowledge from various domains. The ontology can be used to allow agents to share reputation information either directly or indirectly, requiring a semantic mapping process for which the ontology would form an integral part.

At the methodological level, three key contributions have been discussed. The first key contribution is social network analysis. Social network analysis requires the construction of a representation of the relationships within a community. An agent’s position within that representation is indicative of its reputation. Social network analysis goes some way to circumventing the problems associated with acquiring reputation information from other agents. However, social network analysis is not perfect either.
The second contribution is that of relationship analysis. Relationship analysis requires the discovery of relationships among agents given knowledge of their visible environments and their spheres of influence. Once the presence of such relationships has been established they can be clearly identified through a characterisation and classification of relationship types. Identified relationships can then be interpreted in order to adjust the credibility of reputation information.

The third contribution discussed is intended to be used to augment existing trust and reputation evaluation mechanisms. The incentive compatible reputation mechanism provides rational and self interested agents with an incentive to honestly provide reputation information where they otherwise would not do so. The incentive for agents lies in the compensation provided by broker agents that pay for honest and accurate reputation information.

The use of social network analysis in general and relationship analysis in particular, along with the use of reputation ontologies, and an incentive compatible reputation mechanism is of significant relevance. The aforementioned techniques can be adopted and applied to a method or model for trust evaluation with the expectation of promising results. It is expected that the use of the aforementioned techniques will result in more accurate and robust trust models that will conform more closely to the criteria laid out in chapter 4.

Chapter 5 builds on chapter 4 by continuing the investigation into the evaluation of trustworthiness. Therein lies the contribution of chapter 5; further techniques aimed at enhancing agent-oriented trust evaluation have been investigated shedding some light on how the trustworthiness of an agent can be determined.

Chapter 6 departs slightly from the topic of trust evaluation and focuses on agent architectures in a bid to find appropriate architectural models to be used in conjunction with the principles and concepts presented so far.
Chapter 6
Agent Architecture

6.1 Introduction
The focus of the previous chapter was on techniques that can be used to enhance models for evaluating trustworthiness so as to make them more effective. The techniques along with the models and methods presented in chapter 4 provide an overview of the state of the art as far as trust and reputation is concerned. Chapter 3 concentrated on a theory of trust and reputation upon which the scheme for evaluating trustworthiness should be based.

Chapter 6 diverges from the above mentioned topics. The focus of chapter 6 is on potential frameworks or architectures that could serve as a vehicle for agent development. It is expected that by reviewing existing agent architectures, architectural abstractions suitable for developing agents capable of trust evaluation can be identified. Therein lies the relevance of research directed at agent architectures.

Chapter 6 is arranged as follows. Section 6.2 provides an overview of agent architecture. Subsequent sections focus on specific architectural models. Each model is considered critically. Finally the chapter is concluded, further motivating the relevance of the content of chapter 6.

6.2 Agent Architecture
6.2.1 An Overview
‘An architecture usually includes languages and tools for writing programs, as well as an overall philosophy for how programs can be brought together’ [RUS03]. It is the ‘overall philosophy’ as quoted above that is of interest in chapter 6. The ‘philosophy’ is probably best expressed using an architectural model. Numerous such models appear in the literature [RUS03]. For the purpose of sub-section 6.2.1 it is useful to be able to classify, or categorise agent architecture. However, since a complete classification is perhaps not of any great significance in the context of the dissertation a complete taxonomy is not provided.
Agent architecture can be classified as reactive, deliberative or a hybrid of the two [RUS03]. Reactive architecture implies the agent is able to respond in a reflexive manner. Simple condition-action rule based agents (simple reflex agents) are an example of agents exhibiting reactive architectures. A deliberative architectural model implies that a form of reasoning is applied in order for an agent to reach a decision. The hybrid architectural model specifies a reactive capability and a deliberative one. An agent based on a hybrid architectural model is able to react in real-time when necessary and should be capable of further reasoning in order to reach a decision when appropriate [RUS03]. Hybrid models tend to be layered models. According to Russell et al the most popular hybrid model is the three layer architecture [RUS03]. However, other multi-layer hybrid models exist; one such example looks at endowing agents with knowledge of philosophical principles and societal norms [ROS02].

The above classification could be considered very broad; it is based on how an agent responds. Another potential way in which to classify agent architecture is to look at the underlying principles of any architectural model. There is for example the so-called cognitive architecture. Cognitive architectural models attempt to model human reasoning [RUS03]. Examples of architectural models that attempt to model human reasoning include the well established ‘State, Operator, and Result’ architectural model – simply known as ‘Soar’.

According to Georgeff, Pell, Pollack, Tambe and Wooldridge there are also architectural models for practical reasoning agents in which agents strive to maximise utility. The underlying principles of such models tend to stem from economics [GEO98].

Another class of architecture can be identified. Membership of this class is dependant on the explicit modelling of beliefs, desires and intentions (BDI). BDI-theoretic architectural models have BDI principles at their core. BDI-theoretic architecture is well established and probably the most common agent architecture. However, BDI-theoretic architectural models are quite general; it is possible to further differentiate amongst BDI-theoretic architectures although they are all based on BDI principles [GEO98].
As mentioned BDI-theoretic architectural models are well established and quite prevalent. Therefore it is clearly worthy of study. BDI-theoretic architectural models are the topic of the remaining sections of chapter 6. Subsequent sections discuss BDI theory in general before focusing on specific architectural models of interest.

### 6.2.2 BDI-Theoretic Architecture

A Belief, desire and intention (BDI) based architectural model is one that pays specific attention to the modelling of an agent’s beliefs, desires and intentions. An agent’s beliefs refer to what the agent believes about itself and its environment. Beliefs can be considered a means to represent the state of the agent’s world. Beliefs are important in dynamic environments where the agent needs to remember the recent past. Beliefs are also important when the agent’s environment is partially observable; it is sometimes useful for the agent to remember events that occur outside its viewable environment [GEO98].

The desires or goals of an agent relate to what the agent wants and tries to achieve (a desirable end state). A goal driven approach is useful in developing intelligent software. To see why, consider the more traditional task driven approach to software development [GEO98]. The nature of such an approach makes it unsuitable for spontaneously taking advantage of opportunities in the environment. Clearly a task driven approach inhibits proactive and autonomous behaviour.

The intentions of an agent refer to the agent’s chosen plan. The BDI agent’s set of plans constitute the agent’s procedural knowledge [BUS98]. The intention of an agent can also be seen as state information in the sense that it specifies what the agent is currently doing. There is also a school of thought that claims intentions are used to focus reasoning [GEO98]. If the view that intentions are useful for focusing reasoning is accepted, then intentions are also important for constraining the number of possibilities about which an agent must reason.

The above mentioned components allow an agent to be aware; meaning an agent knows what it has done, what it is doing, and what it wants to achieve. The awareness of BDI-based agents means that BDI-based agents have the potential to be reactive, proactive and autonomous. Clearly a BDI-theoretic approach cannot guarantee that
they will be reactive, proactive and autonomous; those qualities are obviously dependant on more than the underlying philosophy or architecture. Nevertheless, a BDI-theoretic approach is a good starting point. Indeed Huber believes it to be a stronger starting point than other architectural models [HUB99].

Although it is suggested by Busetta and Ramamohanarao that BDI agents are in general reactive and social [BUS98], the extent to which they are sociable is questionable. At the architectural level it would appear that (in general) BDI-theoretic architectural models do not explicitly address social ability. The lack of attention to social ability is a concern that is also expressed by Georgeff et al [GEO98].

Indeed the BDI-theoretic family of architectural models has limitations and imperfections (it does not explicitly address social ability) nonetheless it is a strong starting point for developing intelligent agents. The subsequent sections will look at various BDI-theoretic architectural models and their limitations.

### 6.2.3 Summary

Section 6.2 provides an overview of agent architecture. Agent architecture can be classified according to how the agent responds to events in the environment. It is possible to distinguish between reactive, deliberative and hybrid architecture. Agent architecture can also be classified according to the principles behind the architectural model. The BDI-theoretic class of architectural models all model beliefs, desires and intentions, meaning that BDI agents are aware in the sense that they know what they have done, what they are doing and what they want to do. The following sections discuss a few specific BDI-theoretic architectural models.

### 6.3 Procedural Reasoning System Architectural Model

#### 6.3.1 An Overview

The architecture of the Procedural Reasoning System (PRS) is typical of BDI-theoretic architecture; it supports the explicit modelling of beliefs, goals, intentions and plans. Figure 6.1 (adapted from [BUS98]) provides an architectural model of the PRS.
The structure of the PRS incorporates the main elements of BDI-theoretic architecture [BUS98]. Beliefs are typically implemented using records in a database. There is also an intentions structure representing what the agent intends to do next. Additionally there is a goal store and a set of plans. The interpreter acts as the agent’s brain and is responsible for deciding which goals to pursue based on the beliefs. The work of the interpreter involves the selection of a plan that the agent intends to execute in order to achieve the goal(s).

**Figure 6.1 The PRS Architectural Model (adapted from [BUS98])**

### 6.3.2 Analysis

The PRS architectural model is clearly a deliberative one. Therefore there is limited potential for real-time reactivity. The limited potential for real-time reactivity could limit its reusability and thus its generality in situations where real-time reactivity is necessary. There is also little explicit attention given to social behavioural aspects at the architectural level. The lack of attention to social aspects could potentially hinder an agent’s social ability [GEO98].
On the other hand the PRS architectural model is quite simple. Simplicity in itself can be seen as a desirable property. The perceived simplicity is in part due to the seemingly large grain components. If computationally expensive operations are contained within the large grain components performance should be improved [SOM01].

6.3.3 Summary
Section 6.3 introduces what is quite possibly the most well known BDI-theoretic architectural model. The PRS architectural model is a simple, deliberative model. It explicitly models beliefs, desire intentions and plans. An interpreter or reasoning module is also specified. The following section looks at another architectural model that is closely related to the PRS architectural model.

6.4 JAM Architectural Model
6.4.1 An Overview
JAM is a BDI-theoretic mobile agent architectural model proposed by Huber [HUB99]. Not surprisingly, it is quite similar to the PRS architectural model presented above as depicted in figure 6.2 (adapted from [HUB99]).

Much like the PRS architectural model, the JAM architectural model specifies a world model which is a database containing the agent’s beliefs about the environment [HUB99]. The intention structure is also similar in that it is an internal representation of the agent’s intentions relative to the goal being pursued [HUB99]. The architecture also specifies an agent interpreter serving the same function as the one in the PRS architectural model [HUB99]. Similarly there is also a set of plans that the agent may choose to execute. The JAM architectural model also specifies an observer, the purpose of which is to provide the agent programmer with greater flexibility by allowing them to specify the performance of lightweight tasks in between the execution of an agent’s intentions (for example checking a buffer for new messages) [HUB99].

---

5 Large grain components refer to components encapsulating substantial functionality as opposed to fine grain components that encapsulate minimal functionality [SOM01].
Although there is no architectural support for mobility at the architectural level (there is no specification of a mobility component in the model), the architecture does specify how mobility can be achieved through a checkpointing mechanism\(^6\). To achieve agent mobility a plan containing a primitive function is required. The primitive function is responsible for the checkpointing and subsequent migration of the agent to a remote host [HUB99].

![Figure 6.2 The JAM Architectural Model (adapted from [HUB99])](image)

### 6.4.2 Analysis

Since the JAM architectural model is quite similar to the PRS architectural model it is also subject to much the same acclaim and criticism. Therefore, the reader may refer to section 6.3.2 for an applicable critique.

---

\(^6\) ‘Checkpointing’ is discussed in chapter 7
It is worth focusing on the JAM architectural model’s support for mobility. Traditional BDI-theoretic architecture such as that of the PRS do not offer support for agent mobility. Hence there have been efforts to extend the current BDI paradigm to include explicit support for agent mobility [BUS98, HUB99]. While attempts to cater for agent mobility are commendable, it can be argued that the JAM architectural model does not offer optimal support for mobility in terms of architectural abstractions. As mentioned in section 6.4.1 mobility is achieved by an agent selecting and executing a plan containing a primitive migration function [HUB99]. The execution of the plan containing the primitive migration function is useful if an agent proactively decides to migrate to a remote host. But what if the agent host needs to terminate execution suddenly and unexpectedly? It must be remembered that the JAM architectural model is fundamentally a deliberative one. Therefore there is little or no support at the architectural level for real-time reactive behaviour. The lack of support for real-time reactive behaviour is a weakness in the case of events occurring in the environment that require real-time reactive behaviour.

**6.4.3 Summary**

Section 6.4 discusses the JAM mobile agent architectural model, another example of BDI-theoretic architecture similar to the PRS architectural model discussed previously. Perhaps the biggest difference in terms of architectural components is the inclusion of an observer. The observer ultimately serves as an architectural hook that improves flexibility [HUB99]. Mobility is supported through the specification of a primitive function included in a plan. However, the mobility support is perhaps marginally inadequate in situations requiring real-time reactivity given the deliberative nature of the architectural model.

The next section focuses on a more mature BDI-theoretic architectural model that is aimed at providing reasoning agents suitable for Semantic Web applications.

---

7 See chapter 2 for a discussion on the advantages of mobile agents. Further advantages can also be found in [BUS98].
6.5 Nuin Agent Architecture

6.5.1 The Nuin Agent Architecture in Context

The Nuin agent architectural model was developed for the purpose of aiding the development of deliberative agents for Semantic Web applications [DIC03]. According to Dickinson and Wooldridge, Nuin was developed to conform to BDI principles [DIC03]. Dickinson et al also wanted the architecture to yield robust agent applications. Dickinson et al also wanted an architecture that resulted in embedded agents in end-user applications as opposed to the end-user application being embedded in an agent framework. Furthermore, the researchers chose against reinventing the proverbial wheel and instead opted to re-use existing tools and methods where appropriate. Also, since the overall aim was to develop agents for the Semantic Web they opted to make use of emerging Semantic Web technologies [DIC03].

The above paragraph presents the context in which the Nuin agent architecture was developed. The following paragraphs briefly discuss the architecture.

6.5.2 An Overview

The Nuin agent architecture, being an instance of BDI-theoretic architecture, consists of common BDI components and more. The most important aspects of the Nuin agent architecture are depicted in figure 6.3 (adapted from [DIC03]).

Nuin agent’s beliefs are modelled as sentences in first order logic. Desires are modelled as collections of first order sentences that represent goal states in the environment. A model of the agent’s commitment to an action represents the agent’s intentions. A Nuin agent also has a limited memory of its most recent percepts. Percepts are delivered by events in the context of the Nuin architecture [DIC03].

The Nuin architecture also specifies a plan library. Plans are constituted by a sequence of action expressions, post conditions and triggering patterns. Action expressions are made up of atomic actions and tests. Atomic actions and tests comprise the action library [DIC03].
As with many BDI-theoretic architectural models, Nuin also specifies an interpreter that acts as the agent’s brain [DIC03]. The interpreter is responsible for determining the agent’s behaviour. The determination of behaviour is based on percepts (thus the interpreter is responsible for handling the events that deliver the percepts), mental states (beliefs, desires and intentions) and plans.

It is possible for Nuin agents to have multiple knowledge bases, or knowledge sources [DIC03]. Multiple knowledge bases allow the agent to distinguish between its knowledge and the knowledge of others. Dickinson et al have primarily opted to take advantage of more powerful first order logic (as opposed to the slightly semantically weaker description logic). Thus the knowledge bases consist of logical sentences made up from a vocabulary of common literals and combined using the normal logical connectives. Each knowledge base is associated with at least one reasoning component – referred to as a reasoner. Reasoners support resource bounded first order logical reasoning. The need for a resource bounded reasoner stems from the choice to
use first order logic [DIC03]. Reasoning in first order logic is in some cases computationally intractable.

The Nuin agent architecture also specifies an agent middleware layer (see figure 6.3). The agent middleware is a layer of abstraction comprising so-called service adaptors [DIC03]. It is via the middleware layer of abstraction (and the underlying services) that percepts from the environment are received by the agent core. Similarly, it is through the middleware layer of abstraction that the agent affects its environment. The middleware layer protects the deliberative core of the agent from the complexities of the various services and infrastructure (such as message passing) that the agent needs. Infrastructural support can be provided by an existing agent platform.

One final topic worthy of discussion is the Nuin agent configuration. Nuin agents can be configured using a Resource Description Framework (RDF) document [DIC03]. The RDF configuration document can be used to configure the agent’s initial knowledge for example. Dickinson et al believe that controlling agent configuration via a single RDF document makes it easier to programmatically create agents for use in applications [DIC03]. Requiring a single RDF configuration document for agent start-up is preferable to being reliant on an agent framework to bootstrap the agent, which would mean attempting to fit the application into the agent framework as opposed to deploying the agent into the application [DIC03].

6.5.3 Analysis
The Nuin agent architectural model is primarily deliberative. However, Nuin is potentially a hybrid model. The interpreter could be designed to check if events from the event queue match certain conditions [DIC03]. If a condition is met then certain primitive actions from the action library can be executed automatically. Therefore Nuin agents have the potential to be more reactive than agents exhibiting other BDI-theoretic architecture. Consequently, the Nuin agent architectural model is arguably more amendable to use in different types of agents and environments.

The architectural model also shows a desirable separation of concerns. The interpreter contains the logic for deciding on an action while the reasoners are for making
inferences [DIC03]. Other services like agent communication are delegated to the existing services provided by the underlying agent framework [DIC03].

Another potential benefit is the separation of belief, the agent’s knowledge and the knowledge of others. Knowledge is what the agent knows to be true. That is to say knowledge is certain and can be used in conjunction with appropriate reasoning techniques to generate sound and complete inferences. Belief on the other hand implies uncertainty and thus is often subject to a different set of reasoning techniques [RUS03].

It is also possible to speculate about the extensibility of a Nuin agent. It is conceivable that additional special purpose reasoners could be added to the agent by modification of the RDF configuration file. Given the configurability via an RDF configuration document [DIC03], it is expected that extensibility is achievable.

Despite the numerous potential strengths of the Nuin architectural model, there is little or no architectural support for social abilities. Therefore it is questionable as to how socially capable a Nuin agent will be in a multi-agent environment.

6.5.4 Summary
Section 6.5 discusses the more mature Nuin agent architectural model. The Nuin architectural model is a BDI-theoretic architectural model but unlike the other examples of BDI-theoretic architecture discussed so far, it is designed for wide-spread use in Semantic Web applications [DIC03]. The Nuin agent architecture explicitly models knowledge bases and reasoners [DIC03]. The developers have opted to use the semantically stronger first order logic as opposed to description logic [DIC03]. The reuse of pre-existing services (agent communication for example) is facilitated through a layer of abstraction. Nuin agents are also RDF configurable [DIC03]. Nuin agents have the potential to be deliberative as well as reactive. The architectural model also separates concerns well. Despite the strengths of the model, there is little support at the architectural level for agent social behaviour.

The following section takes a look at a more modern agent architectural model.
6.6 Modular BDI-Theoretic Architecture

6.6.1 Novak’s Modular BDI Agent Programming Architecture in Context
Novak and Dix propose a BDI theoretic agent programming architecture [NOV06]. Their research acknowledges that for an agent oriented programming language to be widely adopted it must support existing software engineering practices (not necessarily existing software engineering paradigms). Furthermore, Novak et al acknowledge the importance of seamless integration with external software systems [NOV06]. Not surprisingly then, Novak et al focus on the syntax and semantics of an agent programming language. However, Novak et al also specify modular BDI-theoretic architecture which is the topic of the next sub-section [NOV06].

6.6.2 An Overview
Like any BDI-theoretic architectural model, the model presented by Novak et al also models beliefs, desires and intentions [NOV06]. However, it also specifies capabilities as depicted in figure 6.4 (adapted from [NOV06]).

![Figure 6.4 A Modular BDI-Theoretic Architecture (adapted from [NOV06])](image-url)
The modular BDI-theoretic architectural model also specifies that the various components (beliefs, desires, intentions and capabilities) should implement query and update interfaces [NOV06]. Thus, in terms of the internal workings of components, components are largely independent of one another. As a consequence components are plug-able so long as they conform to the requirements of the interfaces.

Agents that conform to the modular BDI-theoretic architecture of figure 6.4 can interact with their environment through the use of capabilities. Capabilities also support the query and update interfaces [NOV06]. By querying capabilities the perceived state of the environment can be determined. Similarly by updating the capabilities, actions are performed.

The interaction rules, as selected and executed by the interpreter, are simply production rules - or condition-action rules. The interaction rules govern the interaction between the various components [NOV06]. An example of such a rule might be: if Query(capabilities, percept $\alpha$) is true then Update(beliefs, $\alpha = T$). In the case of the example interaction rule, $\alpha$ has been perceived and the agent needs to update its beliefs about its environment.

6.6.3 Analysis
From the overview in the above sub-section the simplicity of the model should be evident. The simplicity of the modular BDI-theoretic model is a strength. Furthermore, relatively large grain components are specified. Depending on how capabilities are implemented, computationally expensive operations could be localised. Thus it could be argued that the architectural model promotes performance [SOM01].

Another potential strength is that the architectural model is a borderline hybrid model. The interaction rules can be used to query the capabilities to check for a particular event in the environment. If the condition holds, then the consequent of the interaction rules follows and the capabilities are automatically updated to affect the environment [NOV06].
The specification of interfaces is the salient characteristic [NOV06]. The specification of interfaces, along with the separation of concerns (beliefs, desires, intentions, capabilities), promotes modularity. Modularity makes it easy to modify or replace individual components. Modular architectural models generally help to make software easier to maintain [SOM01]. Moreover, the architectural model is fairly general and thus is particularly reusable in different contexts.

Despite the modularity and generality of the architectural model, the extent to which social behavioural capabilities can be added is questionable. The capabilities abstraction is meant to serve much the same purpose as the monitor and command generator of the PRS [BUS98, NOV06]. Beyond the specification of capabilities there is little or no support at the architectural level for agent social behaviour.

6.6.4 Summary
Section 6.6 briefly summarises a modular, BDI-theoretic architectural model. The main features of the model include the specification of beliefs, desires, intentions and capabilities. Furthermore, query and update interfaces are specified. Interaction rules that rely on the query and update interfaces are selected and executed by the interpreter [NOV06]. The model is simple and efficient. Perhaps the most valuable contribution is the modularity and associated benefits provided by the model. The extent to which socially capable agents could be developed using the modular BDI-theoretic architectural model remains questionable.

6.7 Conclusion
Chapter 6 introduces agent architecture. Section 6.2 provides an overview of agent architecture by informally considering potential criteria for classification. The same section also introduces the BDI paradigm. The following sections each look at an architectural model by providing an overview and a brief analysis of the model. Included in chapter 6 is the common PRS architectural model, the JAM architectural model, Nuin agent architecture and a modular BDI-theoretic agent architecture.

BDI-theoretic architecture has its strengths. Perhaps most importantly BDI-theoretic architecture results in ‘aware’ agents, meaning agents know what they have achieved,
what they are doing, and what they want to achieve. Without the explicit modelling of beliefs, desires and intentions it is difficult to see how aware agents can be produced. Despite all its strengths BDI-theoretic architecture is not without its limitations.

Perhaps the most concerning aspect of the architectural models surveyed in chapter 6 is the lack of support at the architectural level for agent social behaviour. Indeed it is suggested that ‘the basic BDI model gives no architectural consideration to explicitly multi-agent aspects of behaviour’ [GEO98]. The challenge is to extend the basic BDI model to incorporate social aspects of agent behaviour, perhaps through the use of the existing beliefs abstraction and components that embed social capabilities.

The main contribution of chapter 6 is the identification of architectural abstractions (BDI-theoretic architectural abstractions) suitable for the development of rational agents that are capable of reasoning about trustworthiness. In general, BDI-theoretic architectural models share common architectural abstractions that are of interest. Among the abstractions of interest are beliefs, desires, intentions, plans, and actions. Additional abstractions include knowledge bases, interpreters, and controllers. The beliefs abstraction forms the most important architectural abstraction given the decomposition of trust into constituent beliefs as presented in chapter 3. All the identified architectural abstractions will be used in providing a solution that facilitates trust evaluation. The solution will in turn demonstrate how a rational agent can intelligently determine the trustworthiness of another entity.
Chapter 7

Mobile Agent Systems: A Survey

7.1 Introduction

Up until now the research presented in the dissertation has mostly focused on evaluating trustworthiness. The previous chapter is perhaps the exception, the topic of which is existing agent architectures. Chapter 7 diverges from the themes of trust evaluation and agent architecture and instead focuses on mobile agent systems. Mobile agent systems were introduced in chapter 2. Chapter 7 looks at existing mobile agent systems and aims to provide a qualitative comparison and in so doing, establishes if commonalities exist amongst mobile agent systems.

It is important to investigate existing mobile agent systems in order to understand not only the support they offer to agents but also their current limitations. Of particular interest are the mobility support and the agent protection support offered by mobile agent systems. By establishing the general extent of support for mobile agents the current limitations of mobile agent systems are determined. Insight into the limitations of mobile agent systems allows for the identification of potential improvements. Potential improvements to mobile agent systems can translate into requirements for mobile agents (for example the requirement for mobile agents to have a trust evaluation capability) because the responsibility to manage threats in a mobile agent system is a shared one. Kun, Xin, and Dayou motivate the sharing of responsibility to manage threats in their discussion of a bi-directional security mechanism for mobile agent systems [KUN00].

Chapter 7 is arranged as follows: the next section examines various mobile agent systems in terms of the type of mobility supported, how mobility is achieved and the support for agent protection where it exists. Towards the end of the section a functional comparison is drawn between the surveyed mobile agent systems. Section 7.3 asks what makes a good mobile agent system. The result is criteria suitable for a qualitative, non-functional comparison of the investigated mobile agent systems. A non-functional comparison
(different from the functional comparison of section 7.2) is made in section 7.4 using the established criteria as a basis. Finally, the conclusion outlines potential requirements relating to mobility and agent protection for mobile agents.

7.2 Mobile Agent Systems

7.2.1 Mole
The first version of Mole was completed in 1995 at the University of Stuttgart. The researchers responsible for its development claim that it is the first mobile agent system developed in the Java language. Mole is freely available for download [BAU98].

Mole supports only weak migration. Code mobility, including both weak and strong migration was introduced in chapter 2. Suri et al promote the support of strong migration over weak migration [SUR00a, SUR00b]. Contrary to this however, the developers of Mole believe that weak migration is sufficient and even advantageous for reasons specified in the literature [BAU98]. The choice of weak migration means that Mole can run on any normal Java Virtual Machine (JVM) and without the use of a pre-processor\(^8\).

Mobility in Mole is achieved through the use of Java Remote Method Invocation (RMI) and object serialization. After a call to the relevant method, the agent thread is suspended. Once all outstanding messages have been handled the objects comprising the agent are serialised and transmitted to the remote host. On arrival at the remote host the objects are instantiated and the agent thread starts executing. Finally, a success message is sent back to the source location where the existing agent thread is terminated. A more detailed description of mobility in Mole can be found in the literature [BAU98].

In terms of security Mole does provide host protection. Furthermore, the developers of Mole acknowledge the importance of both host and agent protection [BAU98]. However, mobile agent protection support is a difficult problem and as such there is little evidence in the reviewed literature of any mobile agent protection support in Mole.

\(^8\) The most common way to provide strong migration in Java based mobile agent systems is through the provision of a customised JVM that allows for the capture of execution state. Another way of achieving strong migration is by using a pre-processor to add state saving code to the agent [SUR00b].
Sub-section 7.2.1 introduced Mole. The following sub-section looks at another mobile agent system.

### 7.2.2 D’Agents

D’Agents, formerly known as Agent Tcl, was released in 1995 and is freely available for download. Perhaps the most significant distinguishing factor between D’Agents and the other surveyed mobile agent systems is that D’Agents offers more flexibility by giving the developer the choice of writing agents in any of three different languages. The languages include Java, Tcl and Scheme [GRA02, SCH03].

D’Agents supports strong migration [BRA02, GRA02]. Strong migration was achieved by modifying both Tcl and Java interpreters so as to capture the execution state of an agent. The developers motivate the decision to implement strong migration by claiming ‘strong mobility is easier than weak mobility for the agent programmer to understand and use’ [GRA02]. However, the developers also acknowledge that strong migration requires more effort from the system programmer.

To achieve strong migration, the execution state of the thread of the agent that initiated the migration is transferred with the agent. Therefore, in the case of multi-threaded agents, only the execution state of the thread that requested the migration is transferred [GRA02]. As far as the data state is concerned, all data is transferred for Tcl agents whilst only the data reachable from the Java agent object is transferred in the case of Java agents [GRA02]. All the agent code is transferred with the agent, although the developers of D’Agents acknowledge that the caching of agent code at agent hosts will reduce the overhead of agent migration when an agent has already visited a host [GRA02]. Interestingly enough the developers of D’Agents regard the use of the code on demand technique to ‘weaken the mobile agent abstraction’. The code on demand technique can be seen to weaken the mobile agent abstraction because use of the code on demand technique makes the mobile agent dependant on a machine other than the one it is currently located at [GRA02].
The support offered by D’Agents for agent protection is limited to protection during migration across the network [GRA02]. D’Agents actually makes use of the Pretty Good Privacy (PGP) program to ensure confidentiality and integrity of the agent during transmission. The agent code and state is written to a file (precisely how this is done is unclear as mentioned above). D’Agents then calls upon a standalone PGP process to encrypt the file. The encryption is achieved using symmetric encryption (IDEA) with the secret key encrypted using a public key. Both the encrypted file containing the agent and the encrypted key for decryption is transferred to the remote host [GRA98].

Sub-section 7.2.2 surveyed the D’Agents mobile agent system. The subsequent sub-section discusses Aglets – perhaps the most well known mobile agent system.

### 7.2.3 Aglets

With the advent of the Java programming language, IBM’s Tokyo Research Laboratory decided to develop Aglets. The first version of this Java based mobile agent system was introduced in 1996. The Aglets binary code is freely available for download [WON99].

The Aglets mobile agent system supports only weak migration [BRA02]. The Aglets mobile agent system relies on the standard JVM. Mobility is achieved through the use of object serialization and streams. Initially a dispatch method is called. The protocol for transfer, the domain name and the identifier of the Aglet workplace on the host are specified as arguments. The aglet\(^9\) is then serialized and transmitted to the remote host. On arrival it is de-serialized and the aglet is re-instantiated. Certain mobility events occur on dispatching the aglet and on its arrival at the remote host. The programmer can write code to handle such events [LAN98].

There is no evidence that Aglets offers support for agent protection. In fact, it is asserted that while Aglets does offer a degree of server resource protection there is no support for agent protection [KAR98].

---

\(^9\) An aglet is the term given to a mobile agent in the Aglets Mobile Agent System.
Sub-section 7.2.3 discussed the well known Aglets Mobile Agent System. The following sub-section discusses Concordia.

### 7.2.4 Concordia

Concordia was developed and introduced by Mitsubishi in 1998. Concordia is yet another mobile agent system based on Java [KOB99, GRA02]. Concordia’s binary code is freely available.

Concordia supports weak migration and relies on the standard Java virtual machine (JVM) [WON97, GRA02]. Concordia primarily relies on the notion of an agent propagation server to transfer an agent between hosts. In the context of Concordia a propagation server is referred to as a conduit server. When an agent wants to migrate it makes use of the interface provided by the conduit server. The conduit server transfers the agent to another conduit server on a remote host. The transfer is accomplished with the aid of Java’s object serialization mechanism [WON97].

To achieve reliable and efficient transfer the conduit server works in conjunction with the local queue manager. The queue manager is responsible for the reliable transfer of the agent. The reliable transfer of the agent is achieved through the use of agent queues and a handshaking protocol between the local queue manager and a remote queue manager. The queue manager uses the Java RMI to communicate with remote queue managers [WON97].

Mobility in Concordia is managed through the use of an itinerary [WON97, GRA02]. An itinerary is a data structure external to the agent. The itinerary keeps a record of where the agent has been, where it is going, and what it is going to do when it gets there. The itinerary can be modified at runtime. The advantage of such a scheme is that the owners of agents can easily locate their agents.

Another noteworthy mobility related feature of Concordia is that agents have multiple entry points. Multiple entry points means that the agent program can start executing at
one of many methods when it arrives at a remote host. Many other mobile agent systems (most notably Aglets, although Aglets does allow for event handling on arrival) only allow one entry point, from which it needs to be determined what the agent has previously done [WON97]. Using an itinerary and the multiple entry point scheme relieves the agent of the unnecessary burden of establishing what it has previously accomplished. More detailed information on multiple entry points and the above mentioned mobility support can be found in the literature [WON97, KOB99, GRA02].

Concordia does provide a degree of agent protection support. Concordia protects an agent’s machine code and state during transmission by applying the Secure Sockets Layer (SSLv3) protocol. The use of SSLv3 implies the use of both symmetric key (secret key) and asymmetric key (public key) encryption [WON97, KOB99]. Concordia also protects an agent’s ‘on disk representation’ through the use of encryption.

The protection support offered by Concordia does not protect the agent from all malicious host threats and is therefore inadequate. The threats in question include the malicious host interfering with the execution of the agent at runtime.

While Concordia also provides for host protection, the details of host protection are beyond the scope of the dissertation research. Sub-section 7.2.4 discusses the Concordia mobile agent system. The subsequent sub-section focuses on another mobile agent system, namely NOMADS.

7.2.5 NOMADS
A more recently developed mobile agent system is presented by Suri, Bradshaw, Breedy, Groth, Hill, Jeffers, Mitrovich, Pouliot, and Smith [SUR00a, SUR00b]. NOMADS is freely distributable.

The NOMADS mobile agent system is one of the few Java based mobile agent systems that supports strong mobility [BRA02, GRA02]. As a consequence NOMADS employs a custom developed Java compatible virtual machine – AROMA. The AROMA virtual
machine allows for the capturing of the execution state of an agent. NOMADS also supports forced mobility, whereby an agent host can force the agent to move to another remote host [SUR00a, SUR00b].

Strong migration is achieved in NOMADS through the use of a ‘go’ primitive. In the case of strong migration the execution state of the agent has to be captured prior to migration\(^\text{10}\). Once the architecture independent execution state has been captured it becomes possible to transfer the agent and its state to the remote host. There is little information available in the literature as to precisely how transmission is achieved [SUR00a, SUR00b].

The custom virtual machine AROMA offers support for host protection. In fact, providing increased fine grained host protection forms part of the developers’ future work [SUR00a, SUR00b]. There is little evidence of any support for mobile agent protection.

Sub-section 7.2.5 discusses the NOMADS mobile agent system.

### 7.2.6 Ajanta

Ajanta is another Java based mobile agent system. Ajanta was developed at the University of Minnesota and is a more recent research effort with the first version being made available in 1999.

Ajanta offers both mobility support and agent protection support. The former of which is now discussed. Like most Java based mobile agent systems Ajanta supports only weak migration [BRA02]. The use of weak migration is primarily to make use of a standard JVM. The designers of Ajanta also motivate the choice of weak migration by suggesting that programmer specified mobility (the programmer specifies when the agent is allowed to migrate) is sufficient in most cases [TRI02]. Indeed, strong migration is only necessary if the agent must be able to migrate at any point in its execution. Strong migration is particularly useful if fault tolerance is a strict requirement [TRI02].

\(^{10}\) This process is commonly known as ‘checkpointing’ [BRA02, SUR00b].
At the heart of Ajanta’s mobility support is Java’s object serialization mechanism and an Agent Transfer Protocol (ATP). Agent migration is a two phase process [TRI02]. In the first phases the host server and the remote server use the ATP to coordinate the transfer of the agent in advance. In the event that the remote host permits the transfer of the agent the objects comprising the agent are serialised and transferred. On receipt the objects are de-serialised and a new thread is assigned. Any class code that is required can be downloaded on demand from a specified code server. A more detailed explanation of the above process can be found in the literature [TRI02].

In terms of agent protection Ajanta has a fair amount to offer. An agent can be protected during transfer using symmetric encryption (DES) [KAR98]. On the host, agents are executed in isolation from each other and are thus protected from interference by other agents. Further details can be found in the literature [TRI02]. With respect to protection from malicious hosts, Ajanta makes use of public key encryption to ensure privacy and integrity of an agent’s data state [TRI02]. Ajanta agents have a targeted state which means that certain data meant for a specific host is encrypted with that host’s public key. Ajanta agent’s also have read only state, the integrity of which is protected using the agent’s private key. An Ajanta agent also carries an append only container for information gathered at various agent hosts on its itinerary. Tampering with the container can be detected post the event [TRI02].

The agent protection offered by Ajanta is quite significant and without question is an improvement on many other mobile agent systems [KAR98, TRI02]. However it is not by any means a perfect solution; perhaps the biggest criticism is that it is possible for a malicious host to still interfere with the data state of an agent. While such interference may be detected (by checking the integrity of the state) it is not prevented. Thus a non-intelligent agent may keep migrating to the host which could amount to a denial of service attack. Irrespective of whether a denial of service attack occurs, the agent still wastes both time and resources by migrating and executing (without learning from its mistake). Clearly, continual migration to a malicious host is avoidable if the agent concerned has the capability to evaluate the trustworthiness of such a host. It is also
possible that the agent’s unprotected data (which includes the agent base class [TRI02]) could be modified at a malicious host to make the agent malicious.

Sub-section 7.2.6 discussed the mobility support and the agent protection support offered by Ajanta. The agent protection support is in most respects more comprehensive than that offered by the other mobile agent systems surveyed in chapter 7.

7.2.7 Summary
Section 7.2 covered a representative cross section of the many mobile agent systems that have been developed. The systems surveyed include Mole, D’Agents, Aglets, Concordia, Nomads and Ajanta. The survey is by no means a complete analysis of mobile agent systems. Such an analysis is well beyond the scope of the dissertation. Instead section 7.2 has focused specifically on the type of mobility supported and how the mobility is achieved. More importantly, attention has been paid to the agent protection support (if any) offered by each of the surveyed mobile agent systems. Table 7.1 provides a summary of the surveyed mobile agent systems. The subsequent section focuses on potentially salient non-functional properties of mobile agent systems.

7.3 Criteria for Quality Mobile Agent Systems
7.3.1 The Relevance of Criteria
The previous section surveyed various mobile agent systems according to the pertinent functionality in the context of the dissertation. Section 7.3 presents the important non-functional properties of a mobile agent system. Such non-functional properties are relevant for two main reasons. Firstly, the non-functional properties serve as criteria that are the basis for comparing mobile agent systems. The need to compare mobile agent systems arises when the researcher or developer requires a vehicle for development. Secondly, the non-functional properties have some relevance in the context of developing mobile agents. To understand the relevance of the non-functional properties it is important to take a holistic view of mobile agent systems; a mobile agent system could be
<table>
<thead>
<tr>
<th>System</th>
<th>Languages Supported</th>
<th>Type of Mobility</th>
<th>Mobility Support</th>
<th>Agent Protection</th>
<th>Agent Protection Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mole</td>
<td>Java</td>
<td>Weak migration</td>
<td>RMI &amp; object serialization</td>
<td>None</td>
<td>N/A</td>
</tr>
<tr>
<td>D’Agents</td>
<td>Tcl, Java, Scheme</td>
<td>Strong migration (custom interpreters)</td>
<td>File transfer</td>
<td>Limited</td>
<td>Cryptographic transfer (courtesy of PGP)</td>
</tr>
<tr>
<td>Aglets</td>
<td>Java</td>
<td>Weak migration</td>
<td>Object serialization &amp; streams</td>
<td>None</td>
<td>N/A</td>
</tr>
<tr>
<td>Concordia</td>
<td>Java</td>
<td>Weak migration</td>
<td>Object serialisation, propagation servers, &amp; queue manager (RMI).</td>
<td>Limited</td>
<td>Cryptographic transfer &amp; storage</td>
</tr>
<tr>
<td>NOMADS</td>
<td>Java</td>
<td>Strong migration (custom virtual machine)</td>
<td>’go’ primitive, checkpointing used.</td>
<td>None</td>
<td>N/A</td>
</tr>
<tr>
<td>Ajanta</td>
<td>Java</td>
<td>Weak migration</td>
<td>Object serialisation &amp; ATP</td>
<td>Substantial</td>
<td>Cryptographic transfer, encryption for privacy, digital signatures for tamper detection.</td>
</tr>
</tbody>
</table>

Table 7.1 A Summary of Mobile Agent Systems
thought of as comprising of two parts. The first part is the mobile agent infrastructure which is the subject of chapter 7. The second part is the mobile agents themselves. Thus, from a holistic viewpoint, the non-functional properties discussed in section 7.3 can be considered emergent non-functional properties. That is to say they emerge from the totality of a mobile agent system not just one of its parts. Therefore, the non-functional properties need to be taken into consideration in developing mobile agents.

The following sub-section explains how the non-functional properties of the subsequent sub-sections were derived.

7.3.2 Deriving Criteria
The non-functional properties discussed here are partly derived by considering the potential advantages of mobile agency as discussed in chapter 2 and presented by Lange and Oshima [LAN99]. Clearly the more salient properties are those that maximise the benefits of mobile agency. The properties were also partly derived by considering the potential application domains for mobile agent systems. A list of potential application domains can be found in the literature [LAN99].

7.3.3 Secure Mobile Agent Systems
For mobile agent systems to be adopted for use in real world applications where security is a concern, solutions need to be found for security related problems [BOR02]. Agents and/or hosts that are readily interfered with will limit the autonomy of agents. Furthermore, if such agents are compromised (and this is detectable) the agents have to be re-instantiated and transmitted from the owner’s machine across the network to other remote hosts. The retransmission of mobile agents could limit any benefit of using mobile agent systems to reduce network load [LAN99, VIG04]. It is also worth considering the need for relatively secure mobile agent systems if they are to be applied to e-commerce, personal assistance, and secure brokering domains.
7.3.4 Reliable and Robust Mobile Agent Systems
Mobile agent systems also need to be robust and reliable. In order to realise the potential fault-tolerance benefits of mobile agent systems, agents need to be recoverable in the event of unexpected host crashes [LAN99]. Also should a host need to terminate its service agents should be forced to migrate. Such properties are particularly important in telecommunication network service applications for example.

7.3.5 Efficient Mobile Agent Systems
Performance is also a key property. Two potential advantages of mobile agency are the potential to reduce network load and the potential to overcome network latency [LAN99, VIG04]. If mobile agent systems are not efficient or require a significant amount of code and data to be transferred during agent migration then any benefit is minimised.

7.3.6 Summary
Section 7.3.6 discusses the relevance of criteria for comparing mobile agent systems. Attention is also paid to how the non-functional criteria were derived. Four non-functional properties that should be satisfied by a mobile agent system were presented. Mobile agent systems should be secure, should perform well and should be both reliable and robust. The list of properties is not necessarily complete. There may be other properties of importance. However, the identified non-functional properties are believed to be the salient properties. The following section briefly compares the mobile agent systems of section 7.2 according to the identified criteria.

7.4 Comparing Mobile Agent Systems
7.4.1 A Qualitative Comparison
The previous section provided criteria for comparing mobile agent systems. Section 7.4.1 aims to provide a brief qualitative comparison of the infrastructure provided by the mobile agents systems surveyed in section 7.2. Given the lack of comparative data on the systems discussed in section 7.2 and the difficulty of a quantitative analysis, a qualitative comparison is unquestionably a more feasible option. The most appropriate way of comparing the systems mentioned in section 7.2 is in a tabular format. A tabular
comparison is provided in the subsequent sub-section. The penultimate sub-section motivates the comparisons.

7.4.2 Tabular Comparison
Table 7.2 is the main focus of section 7.4.2. Table 7.2 provides a tabular comparison of the mobile agent systems surveyed in section 7.2. The comparison considers each mobile agent system in terms of its features relating to security, performance, reliability and robustness.

7.4.3 Justifying the Comparison
Table 7.2 compares the surveyed mobile agent systems. Section 7.4.3 motivates the comparison.

As far as security is concerned it is important to take into account a mobile agent system’s support for both agent protection and host protection. Mobile agent systems support for both types of protection is reflected in the features column of the table.

In terms of performance there are many factors that could affect performance. However, given the potential advantages of mobile agency (see chapter 2) it is important to consider performance relative to potential advantages. Hence only system (mobile agent system) level factors affecting the performance of migration are considered.

On the topic of robustness and reliability (which are closely related) various factors could come into play, the most significant of which is surely the need for persistence support. Persistence support is important so that in the event a host unexpectedly shuts down, the agents can be recovered.
<table>
<thead>
<tr>
<th>System</th>
<th>Security</th>
<th>Features</th>
<th>Performance</th>
<th>Features</th>
<th>Reliable &amp; Robust</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mole</td>
<td>Limited</td>
<td>Limited host protection (Java sandbox model)</td>
<td>Acceptable</td>
<td>Only essential code migrates. Code on demand used.</td>
<td>Sophisticated</td>
<td>Uses a form of checkpointing.</td>
</tr>
<tr>
<td>D’Agents</td>
<td>Basic</td>
<td>Agents protected in migration. Hosts protected (authentication, access &amp; resource control)</td>
<td>Weak (Improvement Expected)</td>
<td>All code migrates, (Code caching support to be added). Execution state also has to be transferred.</td>
<td>Weak</td>
<td>No persistence support.</td>
</tr>
<tr>
<td>Aglets</td>
<td>Limited</td>
<td>Limited host protection (Java sandbox model)</td>
<td>Weak</td>
<td>All code migrates(^{11}).</td>
<td>Good</td>
<td>Persistence support provided</td>
</tr>
</tbody>
</table>

\(^{11}\) No evidence to the contrary in the literature

Table 7.2 A Qualitative Comparison of Mobile Agent Systems
<table>
<thead>
<tr>
<th>System</th>
<th>Security</th>
<th>Features</th>
<th>Performance</th>
<th>Features</th>
<th>Reliable &amp; Robust</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concordia</td>
<td>Basic to Improving</td>
<td>Agents protected in transfer &amp; storage. Host protection (Concordia Security Manager)</td>
<td>Weak</td>
<td>All code migrates (Although the Queue manager is expected to improve performance)</td>
<td>Good</td>
<td>Queue Manager ensures reliable migration.</td>
</tr>
<tr>
<td>NOMADS</td>
<td>Limited</td>
<td>Host protection (AROMA VM providing improved resource control)</td>
<td>Weak</td>
<td>All code migrates&lt;sup&gt;12&lt;/sup&gt;. Execution state migrates.</td>
<td>Weak</td>
<td>No persistence support.</td>
</tr>
</tbody>
</table>

<sup>12</sup> No evidence to the contrary in literature

Table 7.2 A Qualitative Comparison of Mobile Agent Systems
<table>
<thead>
<tr>
<th>System</th>
<th>Security</th>
<th>Features</th>
<th>Performance</th>
<th>Features</th>
<th>Reliable &amp; Robust</th>
<th>Features</th>
</tr>
</thead>
</table>

Table 7.2 A Qualitative Comparison of Mobile Agent Systems
It can be concluded that of the mobile agent systems surveyed, all target the Java virtual machine (or a modified version of it). There is also a tendency amongst mobile agent systems to support weak migration. As far as security is concerned, all the mobile agent systems considered provide some form of host protection. Mobile agent protection however, is at best limited. It can be further concluded that the above commonalities motivate the need for a general architectural solution to support existing mobile agent protection mechanisms, which can benefit numerous mobile agent systems.

7.4.4 Summary

Section 7.4 presents four criteria or emergent non-functional properties that serve as a basis for comparing the mobile agent systems surveyed in section 7.2. The emergent non-functional properties include security, efficiency, robustness and reliability. The systems are compared qualitatively in the table of sub-section 7.4.2. The main reasons behind the comparisons are presented in the previous sub-section. The following section concludes chapter 7.

7.5 Conclusion

Chapter 7 focuses on what is believed to be a representative cross section of the many mobile agent systems available. Most of the mobile agent systems surveyed are research prototypes. Of particular interest in surveying such mobile agent systems is the support offered for agent protection. Support for agent protection is one of the most challenging problems facing mobile agent development. Also, of interest is the support for mobility and how mobility is achieved.

Clearly none of the systems surveyed provide completely adequate agent protection, although Ajanta does come close [KAR98, TRI02]. It is clear that traditional ‘hard’ security approaches have their limits as far as mobile agent systems are concerned. This is not to say that traditional security approaches do not have a role to play. Existing security mechanisms could be augmented with a ‘softer’ approach. Such an approach would see the agent taking responsibility for its protection. Mobile agent responsibility for protection could be achieved through intelligent mobile agents that
have the capability of evaluating the trustworthiness of various entities including hosts in mobile agent system.

It is clear that mobility support is for the most part adequate. Although an interesting question would be if it is possible to develop a procedure to allow for the migration of agents between different mobile agent systems. The problem of inter-system migration could form part of future research.

Also of relevance in chapter 7 are the criteria that a mobile agent system in its totality should meet. Since mobile agents are a part of any mobile agent system, clearly the criteria need to be considered in designing and developing mobile agents.

However, the main contribution of chapter 7 is the investigation of a representative cross section of mobile agent systems. The investigation includes a comparison, which in turn leads to the establishment of common factors amongst the surveyed mobile agent systems. The common factors motivate the need for a general architectural solution to augment existing mobile agent protection mechanisms, which can benefit numerous mobile agent systems. The proposed general architectural solution is the focus of chapter 8.
Chapter 8
A Trust Enhanced Mobile Agent System Architecture

8.1 Introduction
After an examination of the state of the art (chapters 2 – 7) based on the available literature, the focus turns to designing and implementing a solution aimed at enhancing agent protection with a trust-based approach. The solution also serves to demonstrate how an agent can intelligently evaluate the trustworthiness of various entities. The analysis of the literature as presented up to the end of chapter 7 provides a foundation upon which a solution can be built. Suitable ideas gleaned from the literature manifest themselves in the solution as presented in chapters 8, 9 and 10.

As should be clear from the chapter on mobile agent systems – the support for mobile agent protection is at best limited. Nonetheless, these systems are still of value. Furthermore, it is also desirable to avoid ‘re-inventing the wheel’ where possible. Therefore, instead of developing a new mobile agent system from scratch, an addition to any mobile agent system is proposed. The add-on comes in the form of a specialised trust evaluator agent (hereafter referred to as a trust evaluator agent) that provides a trust evaluation service to other user agents. The trust evaluator agent add-on, as proposed by Pike, Ehlers and Oosthuizen, allows for a trust-based approach to be used in augmenting existing mechanisms for mobile agent protection in any given mobile agent system [PIK07].

Chapter 8 is the first of four chapters that outline the details of a solution. A top down approach is taken whereby an architectural model for a generic trust enhanced mobile agent system is proposed. The architectural model for the generic trust enhanced mobile agent system provides the necessary context that allows for the presentation of the trust evaluator agent in subsequent chapters.
The following section presents the architectural model. Subsequent sections concentrate on the various elements of the architectural model.

### 8.2 Architectural Abstraction of Mobile Agent Systems

#### 8.2.1 Overview

At an abstract architectural level, all mobile agent systems can be viewed as a collection of connected nodes that host agents (mobile or otherwise). The solution proposes the introduction of a trust evaluator agent. This scheme is illustrated in figure 8.1.

![Figure 8.1 Abstract Model for a Generic Trust Enhanced Mobile Agent System](image)

It is possible to classify agents in the system as trust evaluator agents, user agents, and host controller agents. The following sections discuss each type briefly.
**8.2.2 Host Controller Agents**

Host controller agents (human or otherwise) control or own the host. It is a reasonable hypothesis that the host is malicious if and only if the controlling agent is malicious. Therefore the problem of evaluating the host is equivalent to the problem of evaluating the controlling agent. Thus aspects of cognitive trust theory (chapter 3) along with techniques for agent-oriented trust evaluation (chapters 4 and 5) can be applied.

The host machine can be considered a physical extension (or actuator) of the controlling agent. As a consequence, observations of the host machine’s behaviour can be used in determining if the controlling agent is malicious. Thus the need arises for trust evaluators to distinguish between malicious behaviour and unreliable behaviour. This simply means that the trust evaluator agent should be able to attribute the success or failure of the agent execution to either environmental factors or to agent internal factors. The former implies an unreliable service from the host. The latter implies malicious intent from the controlling agent. The role of trust evaluator agents is discussed subsequently.

**8.2.3 Specialised Trust Evaluator Agents**

Trust evaluator agents are intended to provide a specialised trust evaluation service to other agents. However, one might question the role of trust evaluator agents altogether. Surely the addition of trust evaluator agents merely shifts the problem of evaluation of another agent’s trustworthiness to the problem of determining if the trust evaluator agent can be trusted?

Indeed this is so. However, through developing a trust evaluator agent, it becomes possible to extract the most important functionality. The important functionality can then be packaged as reusable components. The components can be used to embed a basic and essential intelligent trust evaluation capability into agents. This component-based scheme means that user agents (section 8.2.4) have the capability to choose a trust evaluator

---

13 This is important in the case of a human controller agent that is removed from the environment and thus has little or no other interactions with agents in the environment.
agent. It is then possible for the user agent to delegate the trust evaluation goal to the selected trust evaluator agent. Precisely why such delegation is necessary is explained in section 8.2.4.

Multiple trust evaluator agents may be deployed both locally and remotely at different hosts. The hosts upon which the trust evaluator agents reside should be trusted (owned by some trusted authority). Furthermore, trust evaluator agents should not migrate to other hosts. These measures will help ensure the integrity of the trust evaluator agents.

Trust evaluator agents may deem each other trustworthy with respect to the task of evaluating trustworthiness. This can then result in a so-called ‘web of trust’ [GUH04]. In this context a web of trust refers to a network of trust evaluator agents that share reputation information. Such a scheme could reduce the burden on any well connected trust evaluator agent. However, such a scheme is only sensible if trust evaluator agents share the same goal. This is certainly only an option in cooperative environments (from the point of view of the trust evaluator agents). Trust evaluator agents may purposefully try to deceive each other in environments where they are in competition with one another (perhaps competing for financial remuneration).

8.2.4 User Agents
User agents are those agents in the system that serve one or other user. These agents typically perform some task in order to achieve a goal on behalf of the human user. Examples of such agents may include an e-commerce agent that tries to find the best price for a certain product from various providers. Such an agent may travel from one provider’s host to another provider’s host in search of the best deal.

The user agent’s main purpose, that is - its goals do not relate to trust evaluation. For example, a user agent’s main purpose might be to find the cheapest airfare offered by various operators for the user. Thus, having to continuously evaluate the trustworthiness of other agents and hosts is an unnecessary burden on the agent and its resources. The
proposed scheme allows the user agent to find a trustworthy trust evaluator agent and then to delegate the (typically secondary or supporting) trust evaluation goal.

8.2.5 Summary
Section 8.2.5 proposes a generic view of mobile agent systems. Such systems can be viewed as connected hosts, controlled by host controller agents, providing execution facilities to user agents. By adding trust evaluator agents to these systems, user agents can delegate the supporting trust evaluation goal once they have found a suitable trust evaluator agent. This reduces the burden on user agents. In order for this to be achieved, however, it is necessary that user agents have a basic intelligent trust evaluation capability. The following section concludes this chapter.

8.3 Conclusion
Chapter 8 looks at mobile agent systems from an abstract architectural perspective. The introduction of trust evaluator agents to provide a specialised trust evaluation service is proposed. Both trust evaluator agents and user agents share a basic embedded trust evaluation capability (chapter 11). User agents will have the capacity to find trustworthy trust evaluator agents via the use of the embedded trust evaluation capability. Once a trustworthy trust evaluator agent has been found the trust evaluation goal can be delegated to the trusted trust evaluator agent. This constitutes a hybrid approach – somewhere between a centralised approach and a decentralised approach. A centralised approach would involve trust evaluator agents that would have to be ‘blindly’ trusted by user agents. User agents would not have a trust evaluation capability. On the other hand a decentralised approach would mean that each user agent would be burdened by the trust evaluation goal. This could be particularly problematic if the agent concerned has to evaluate the trustworthiness of other agents regularly.

Clearly the hybrid approach mitigates the disadvantages of the centralised approach and the decentralised approach. Other benefits of the proposed solution include the continued use of existing mobile agent systems. It makes little sense to ‘re-invent the wheel’ and develop a new mobile agent system when so many useful systems exist. These existing
systems can be easily upgraded by adding trust evaluator agents and embedding the basic intelligent trust evaluation capability into user agents. The reusability benefits of components and component-based architecture also apply.

The ultimate benefit perhaps lies in using this solution to augment existing approaches to mobile agent protection. This will reduce risks for mobile agents, and hopefully ensure better interactions among agents.

Chapter 8 has provided a context in which to discuss the trust evaluator agent. The following chapter looks at the trust evaluator agent in more detail. This includes an overview of the architectural model and a discussion of the most important elements.
Chapter 9
Specialised Trust Evaluator Agents

9.1 Introduction
The previous chapter introduced trust evaluator agents as add-ons to mobile agent systems. Trust evaluator agents are intended to provide specialised trust evaluation services to user agents.

Chapter 9 discusses the high-level design of trust evaluator agents. An agent architectural model is proposed and each element of that model is briefly discussed. The specification of the architectural model is merely the application of existing concepts presented in chapter 6. Hence the contents of chapter 9 will seem familiar to the reader. Nonetheless, the contents provide the essential context for the work that follows in chapter 10.

The following section provides an extensive overview of the model. The main elements of the model are also discussed. The specified architectural model is analysed in the conclusion.

9.2 Specialised Trust Evaluator Agent Architectural Model
9.2.1 Overview
Figure 9.1 provides an overview of the architecture of the trust evaluator agent. It is clear that the figure represents a BDI-theoretic architectural model. A BDI-theoretic architectural model was chosen because BDI-theoretic architectures are well understood and commonly used in the development of agents. It is further hoped that this will ease any implementation effort on the part of third parties. For example, it is quite possible that the mobile agents forming part of mobile agent systems exhibit BDI-theoretic architecture. In this case, reuse of existing architecture (with some slight modification) and software would more than likely be possible.
Figure 9.1 Specialised Trust Evaluator Agent Architectural Model
The abstractions offered by BDI-theoretic architecture are also appropriate for managing the complexity of the problem at hand. Furthermore, BDI-theoretic architecture results in ‘aware’ agents – that is agents that know what they believe, what they desire and what they intend. This is consistent with the cognitive trust theory\textsuperscript{14} proposed by Castelfranchi et al [CAS98] and summarised in chapter 3.

The trust evaluator agent is to be embedded in an environment. The trust evaluator agent’s environment is any mobile agent system that can be modelled by the abstract architecture of the trust enhanced mobile agent system proposed in chapter 8. The trust evaluator agent receives percepts from the environment. The trust evaluator agent also acts in its environment through actuators [RUS03]. The agent has an interpreter to interpret percepts. The architectural model also specifies a controller, performance evaluator, learning element and multiple trust evaluation components. The model also distinguishes between knowledge and belief. The distinction between knowledge (what the agent knows to be true) and beliefs (what the agent believes to be true) is important because beliefs imply uncertainty and thus a different set of reasoning techniques are required.

The following sections discuss each element of the above architectural model.

\textbf{9.2.2 Sensors and Actuators}

Sensors and actuators are responsible for perceiving and acting upon the environment respectively [RUS03]. At an abstract level, the sensors must receive the raw perceptual information and forward it to the interpreter. Since trust is a social phenomenon, it is foreseeable that most percepts received will be communicative messages from other agents. It is also possible that the agent may read records from a log database, in order to access information of past interactions for example.

\textsuperscript{14} That cognitive trust theory claims that only a cognitive agent is capable of trusting. A cognitive agent is clearly an agent that is aware of its beliefs and can reason about them.
Actuators execute the atomic actions that affect the environment. The set of atomic actions defines the agent’s so called ‘sphere of influence’. The actuators receive messages from the controller to execute a specified atomic action. Again, since trust is a social phenomenon and the trust evaluator agent is providing a trust evaluation service, it is likely that most actions will involve transmitting communicative messages to other agents.

9.2.3 Interpreter
The interpreter receives the raw perceptual input from the sensor and processes it. The ultimate role of the interpreter is to interpret the percepts and to update the agent’s knowledge based on the contents of the percepts. Since the knowledge relates to the environment the variables comprising a state description will need to be modified as a result of the interpreter’s work.

9.2.4 Knowledge
The trust evaluator agent has two distinct types of knowledge: descriptive knowledge, and procedural knowledge. The procedural knowledge referred to here is the agent’s plans. Procedural knowledge is discussed later in this section.

The descriptive knowledge can take the form of an ontology. The use of an ontology can provide the agent with both domain specific knowledge and task related knowledge. An example of the use of the latter is illustrated in figure 5.2 (derived from Casare et al [CAS05]) of chapter 5. Another example is the possible use of an ontology by the interpreter for interpreting messages received from other agents.

Descriptive knowledge may also take the form of objects in memory stating the truth or falsehood of some fact (where a fact is actually a relationship among members of sets for example). These objects can be classified as relating to the other agents, or purely the environment. Such knowledge can be stored persistently as information, meta-data and meta-information in a database.

144
9.2.5 Beliefs

The model also specifies that the agent should have beliefs. Beliefs are distinctly different from the knowledge specified above. As mentioned, beliefs imply uncertainty and as such are subject to a different set of reasoning techniques.

The agent may hold beliefs about other agents as well as its environment. The beliefs relating to the environment influence the agent’s evaluation of the trustworthiness of the environment. Similarly, the beliefs relating to other agents influence any evaluation of the trustworthiness of those agents.

Beliefs can be represented by objects in memory encapsulating the probability of a fact. Where there are relationships (particularly dependence relationships) between beliefs, the corresponding objects can be connected to one another via the edges in a graph structure. A graph of beliefs is potentially useful as it allows for the application of searching techniques to answer queries relating to the agent’s beliefs. Admittedly, due to the memory costs involved, creating a belief network only makes sense if the trust evaluator agent has numerous related beliefs, where the relationships between beliefs are complex and may change at runtime.

If persistent storage of beliefs is required then they must be stored as information in a database.

9.2.6 Desires and Intentions

An agent’s desires are indicative of all the things that the agent could possibly want to achieve. Desires can be thought of as a library of goals where each goal is a state description describing a desirable state in the environment.

If goals are to be designed according to what the designer actually wants in the environment, then the trust evaluator agent’s primary goal would be to minimise incidents of betrayal in the environment. A more task oriented definition of the trust evaluator agent’s primary goal is to provide accurate trust evaluation reports (reputation information) to agents that request the service they provide.
Intentions represent what the agent has chosen to achieve. Intentions are formed by deliberation and amount to a selected goal that the agent will persist to achieve.

9.2.7 Plan Library
The plan library contains the agent’s procedural knowledge. A plan library is a collection of plans. A plan in this case is considered as a specification of a sequence of activities. An activity can be considered as a plan, a process of deliberation or an atomic action. The specified sequence of activities constitutes the body of the plan. A plan has both pre-conditions and post-conditions. A pre-condition is a specification of state (state description) that must hold prior to the plan being executed. Similarly, a post-condition is a specification of state (state description) that must hold after the plan has been executed.

9.2.8 Controller
The controller controls the agent’s execution. The controller is responsible for selecting an intention; selecting a plan that will achieve the intention; bootstrapping other deliberative processes; and invoking the actuator to execute atomic actions.

The controller will determine if the agent’s intention has been realised. If the intention has been realised, or if it is no longer possible or desirable to realise the intention then the agent needs a new intention. Through a process of deliberation, a new intention is selected [WOO00]. In practice, this involves the controller selecting and filtering the desires to determine the new intention.

Once a new intention has been formed the controller then needs to select a plan from the plan library in order to realise the intention. This is achieved via a process of ‘means-ends reasoning’ [WOO00].

Once a plan has been selected, the controller will either bootstrap a deliberative process (in the event of a deliberative activity) or invoke the actuator to execute an atomic action (in the event of an action activity).
Suitable algorithms for implementing the controller element of this model can be found in the literature [WOO00].

9.2.9 Trust Evaluation Components
The trust evaluation components referred to in the architectural model are intended to be ‘pluggable’ components that embed some specialised capability. As implied, there may be multiple components – each providing a specific deliberative capability.

For example, one component may provide for the actual process of evaluation. Another component may cater for the handling of reputation information. Yet another component may provide a specialised relationship analysis capability.

9.2.10 Performance Evaluator
The performance evaluator monitors the agent’s performance. The performance evaluator includes a utility function (discussed in the next paragraph). The performance evaluator requires a state description of the current state from the controller. This state description is passed as an argument to the utility function so that the performance evaluator can determine the utility of the current state. Obviously the determination of utility is only required in complex environments where there are many or possible infinitely many different states. In simpler environments with a small finite number of distinct states, the utility of each state can be specified in advance. In such cases there is no need for a utility function and the performance evaluator need only determine if the agent is performing worse than expected. In the eventuality where the agent is performing worse than expected the performance evaluator can inform the learning element.

The purpose of the utility function is to determine the utility of the current state for the agent. In cases where the current state is the goal state this is trivial (because utility can be specified as part of the goal state description beforehand). In cases where the current state is distinct from the goal state, utility will have to be calculated. One possibility is to determine how close the current state is to a goal state by considering how many variables in the current state description match the relevant variables in the goal state
description. Utility is required if a decision-theoretic approach to uncertain reasoning is adopted. Clearly there must be some relation between the agent’s desires and intentions on one hand and utility on the other. Intuitively, states where the agent has realised its intentions should yield the highest utility.

### 9.2.11 Learning Element

The purpose of the learning element is to make improvements to the various structures that may exist. For example the learning element may be able to re-structure a Bayesian network (in the hypothetical situation that one is used) to reflect new causal knowledge.

The importance of a learning element in producing reactive, proactive, social and particularly autonomous agents is acknowledged [RUS03]. Hence, a learning element has been included in this model. However, the precise details of such a learning element are beyond the scope of this work. Therefore the learning element is not considered further in the context of this work.

### 9.3 Conclusion

Chapter 9 outlines a suitable architectural model for trust evaluator agents. An overview of the model was provided and each element was subsequently discussed. BDI-theoretic architecture is appropriate because it results in ‘aware’ agents. Furthermore, it is appropriate given the cognitive trust theory by Castelfranchi et al [CAS98]. Additionally, BDI-theoretic architecture is well established and understood.

The trust evaluator agent architectural model proposed in chapter 9 is particularly appropriate for deliberative agents. Clearly, a trust evaluator agent can be described as deliberative. However, it is nonetheless desirable to strike the right balance between purely deliberative agents on the one hand, and purely reactive agents on the other [RUS03]. The trust evaluator agent architecture does allow for a degree of reactivity; the controller could be programmed to automatically invoke the actuator given some interpreted percept without the need for extensive deliberation. As a deliberative agent,
with intentions and plans by which those intentions can be achieved, proactive behaviour should be possible.

The social ability of the trust evaluator agent is not directly supported by a dedicated architectural element (a social element that deals specifically with agent interaction). However, social ability is an emergent property of the implementation of the trust evaluator agent architectural model. The trust evaluator agents’ sensors and actuators must support the sending and receiving of messages to and from other agents. Also, the agent should have descriptive knowledge in the form of an ontology to ensure a shared semantics for effective communication. Similarly, the agent’s beliefs relating to other agents should contribute to its social ability.

A deliberative agent has the capacity for autonomy. Certainly, an agent that knows what it wants and how to achieve it is more likely to be autonomous. Thus the specified trust evaluator agent architectural model should aid the autonomy of the trust evaluator agent. The specification of a learning element is also aimed at improving the autonomy of the trust evaluator agent.

The following chapter builds on chapter 9 by focusing on the desires and particularly the plans of trust evaluator agents. Of particular importance is the specification of plans to minimise incidents of betrayal in the environment.
Chapter 10
Desires, Intentions and Plans

10.1 Introduction
Chapter 8 introduced the concept of the trust evaluator agent in the context of mobile agent systems. Chapter 9 proposed an architectural model for trust evaluator agents. This chapter builds on the previous work by focusing more precisely on how trust evaluator agents provide a trust evaluation service to user agents.

Desires, intentions and plans were considered at a high level in the previous chapter. However, the content of the desires, intention, and plans in the context of the trust evaluation problem needs to be taken into account. Chapter 10 therefore looks at the possible desires that translate into intentions at runtime as well as the plans by which the desires can be achieved. The focus of this chapter is predominantly on the content of the BDI abstractions. Representational issues are deferred to chapter 12 on trust evaluator agent implementation. While representation of the abstractions and content is important, it is not the focus of this research especially considering the generic nature of the work.

The following section elaborates on the content of the trust evaluator agent’s desires. Section 10.3 deals with the particulars of plans and plan selection. The plans discussed can be selected by the agent to achieve its intentions. Chapter 10 further provides the context that allows for the presentation of the trust evaluation component in the next chapter.

10.2 Desires and Intentions
10.2.1 A Primary High-Level Goal
As suggested previously, the purpose of introducing trust evaluator agents is to minimise incidents of betrayal in the environment. Therefore, minimising incidents of betrayal in the environment is a goal that a trust evaluator agent should strive to achieve.
10.2.2 Goal Decomposition
It is possible to decompose the above mentioned goal into supporting sub-goals. The achievement of the supporting sub-goals would then result in the achievement of the primary goal. For example one such supporting sub-goal would be for the agent to strive to be a competent and authoritative evaluator of trust. This may seem an obvious and natural enough goal for the agent to pursue. However, it is questionable as to whether or not it is a good design choice. The following sub-section elaborates.

10.2.3 Analysing Goals
In off-the-topic literature, it is common to encounter so-called SMART criteria for business and personal goals [SCH04]. It is worth adapting these criteria for evaluating the agent’s goals. An agent’s goals should be simple, measurable, attainable (most of the time), realistic and time-framed. Given the criteria the above sub-goal can easily be evaluated.

There could be several problems. The goal is not simple to achieve - especially not in open environments. The measurability is even more problematic. How can it be determined if the agent is performing competently? Competence or authority should be judged relative to other trust evaluator agents’ performances. Judging competence requires knowledge of how other trust evaluator agents perform. In the likely event that such knowledge is unavailable the designer may resort to specifying some arbitrary benchmark. A similar problem arises as far as time-frame is concerned; how long should it take for the agent to become competent?

Another problem with decomposing the agent’s primary goal is that unnecessary complexity is introduced. Furthermore, if the decomposition of goals is applied recursively the sub-goals become very task oriented. This is undesirable because it limits the agent’s autonomy and is unsuitable for the agent paradigm.

Therefore the trust evaluator agent’s only goal is to minimise incidents of betrayal in the environment. The goal of minimising incidents of betrayal is no more complex than the goal of being a competent evaluator of trust. Furthermore, the goal is attainable and realistic. As far as time frame is concerned; the agent simply tries to minimise incidents of betrayal at each and every time interval. The goal is also
measurable. An associated performance measure could be the total number of incidents of betrayal reported in the environment for a given time interval. The agent will derive higher utility from states where this number is very low.

One immediate benefit of a single goal is that it allows for a simplification of the algorithms presented by Wooldrigde, since the agent does not have to select an intention from a list of possible desires [WOO00]. The abstraction of desires is nonetheless an important one and should be part of the architectural model if for no other reason other than the extensibility and reusability of the model.

10.2.4 Summary
Section 10.2 briefly looks at the design of a goal (and its performance measure) for the trust evaluator agent. The agent’s goal is to minimise incidents of betrayal in the environment, for any given time interval. The possibility of decomposing the goal to minimise incidents of betrayal in the environment into supporting sub-goals was considered and rejected. This is motivated in the analysis of the previous section. That analysis also explains why the chosen goal is an appropriate one.

As the above paragraph explains, chapter 10 has focused on goals and intentions up until this point. The remainder of the chapter attends to the agent’s procedural knowledge – its plans. These are the plans that the agent can use to achieve its goal and pursue higher utility.

10.3 Plans to Minimise Betrayal
10.3.1 Overview
Section 10.3 specifies potential plans that the trust evaluator agent may select in order to achieve its intention of minimising incidents of betrayal in the environment.

The trust evaluator agent relies on several plans that bootstrap the deliberative processes as provided by the components. Consequentially, the execution of the plans results in belief revision. Belief revision will allow the agent to reach a more informed evaluation. The inclusion of several distinct plans and associated components also allow for the incorporation and consolidation of methods and techniques presented
throughout the dissertation. The plans include a plan for attributing the outcome of delegation; a plan for analysing relationships; a plan for processing reputation information; and a plan for evaluating trustworthiness.

Before delving into the details of the aforementioned plans, the structure of each plan is addressed in the following sub-section. The selection of plans also receives attention in its own sub-section. Each of the remaining sub-sections deal with the specifics of a particular plan.

### 10.3.2 Plan Structure

Each plan has a precondition, post-condition, and a body. Preconditions and post-conditions are assertions in the form of state descriptions. A precondition is therefore a specification of a state that should exist for the plan to be executed. If the plan is selected and executed given the existence of the state conforming to the precondition, then a state conforming to the post-condition will result. The body of the plan is the specification of a sequence of activities. An activity may refer to a sub-plan; a deliberation (reasoning process); or an atomic action.

### 10.3.3 Plan Selection

The agent selects a plan on the basis of the intention and current state description. Since there is only one desire and therefore one intention, it is clear that all plans relate to the achievement of that intention. Thus the agent only has to consider the state description when choosing plans.

### 10.3.4 Trust Evaluation Meta-Plan

Many of the plans discussed in section 10.3 allow the agent to revise its beliefs so as to improve trust evaluation. This in turn should result in fewer incidents of betrayal in the environment. However, these plans need to be orchestrated in such a way so that accurate evaluations result. Figure 10.4 represents a meta-plan for trust evaluation. It is important to note that the figure shows how the plans discussed in this section as a whole fit together. Thus, some overlap is visible between the diagrams in this section and the figure concerned.
In cases where a client agent requests an evaluation of the trustworthiness of a target agent with respect to a particular task for the first time, the trust evaluator agent will need to interrogate its belief sources before proceeding with the actual evaluation. In such cases it is assumed that the client agent has never delegated a task or goal to the target agent before. Therefore, there is no need to detect betrayal. However, reputation information and relationship analysis can be used to establish the strengths of the trust evaluator agent’s beliefs. In this case the plans discussed in subsequent sub-sections apply.

Figure 10.1 Trust Evaluation Meta-Plan
Unfortunately interrogating all sources of belief is likely to be a time consuming process. One possibility to speed up the first evaluation is for the trust evaluator agent to base its evaluation of the target only on reputation. The plan for gathering reputation information is less complicated than the other plans. Therefore the plan is expected to be the least time-consuming, assuming reputation information is readily available (other trust evaluator agents have evaluated the target at some point). However, this is a risky strategy; especially if other trust evaluator agents in the system are incompetent. If the other reputation providing agents are incompetent with respect to trust evaluation executing such a strategy could result in the client agent perceiving the trust evaluator agent as incompetent.

It may be rational for the trust evaluator agent to use only reputation information for first time evaluations if for example the client agent is in a hurry to capitalise on some time-limited opportunity in the environment. Direct experience and the relationships with other agents can be established at a later point in time when the trust evaluator agent is not busy handling any requests.

In subsequent evaluations of the same agent with respect to the same task or goal for the same client agent, it is desirable to include the client agent’s perceptions of the previous delegation experience. The feedback provided needs to be analysed; it is necessary to attribute the success or failure of the delegated task or goal to environmental factors or target agent internal factors. The plan for handling feedback is presented in sub-section 10.3.8.

In subsequent evaluations it may also be necessary to re-interrogate other belief sources. Re-interrogation of the other belief sources may be warranted because the certainty of the trust evaluator agent’s beliefs should decay with the passage of time. Such an assertion reflects the fact that as the agents and their environment evolves, so will their beliefs. Given knowledge of a changing world, the agent should question the validity of beliefs formed in the distant past.

Once the trust evaluator agent’s beliefs have been revised as necessary, it is possible to compute an estimate of the trust evaluator agent’s trust that the goal to be delegated will be achieved successfully. The result of the computation is mapped to a set and
then described using an ontological term. The resulting term is communicated to the client agent. If the client agent is in fact a peer trust evaluator agent that has requested a detailed response, the trust evaluator agent could also reveal its beliefs relating to the target agent. This part of the plan is described in full in section 10.3.9 dealing with the plan for evaluating trustworthiness.

Once the plans that comprise the meta-plan have been executed, the request for an evaluation will have been handled. Unless another request arrived while the agent was performing the evaluation, the trust evaluator agent will be free to select another plan. In general, after communicating the result of the evaluation it is the case that the trust evaluator agent has done everything within its sphere of influence to minimise incidents of betrayal.

10.3.5 Establishing Direct Experience

One source of trust is direct experience. Thus it seems sensible to consider a potential plan that allows the trust evaluator agent to establish direct experience of its own. Such a plan also ensures that the trust evaluator agent is busy with a productive task even if the trust evaluator agent has no immediate requests for reputation information.

Selection and execution of the plan to establish direct experience is conditional on the existence of at least one suitable target agent. A suitable target agent is a host controller agent or user agent (only in the event that the plan was generalised to cater for user agents) that a client agent has requested an evaluation of – at any point in the past. This precondition is important because in many cases it would not be sensible for the trust evaluator agent to randomly select any agent for evaluation. There is no guarantee (especially in larger and open environments) that any client agent will ever request an evaluation of that evaluated agent. Such situations could lead to significant wasted effort on the part of the trust evaluator agent.

Figure 10.2 below provides an overview of the key activities that comprise the plan for establishing direct experience of a target host controller agent. As indicated in the previous paragraph, the selection of a target agent is the starting point. A straightforward approach for selection would be to randomly choose an agent. Other options include choosing an agent based on the time elapsed since the last experience,
or choosing an agent based on the frequency that an agent is the target of evaluation requests. The selection of a target agent constitutes deliberation.

The next step is the action of spawning and dispatching a so-called ‘dummy’ agent. In this step the trust evaluator agent constructs a simple dummy agent with a fake data state. The behaviour of the agent can be provided by specifying the inclusion of components that allow the agent to perform relevant tasks. The dummy agent then migrates to the remote agent host corresponding with the host controller agent for execution.

Spawning dummy agents is based on the assumption that the trust evaluator agent has a library of components with which to build dummy agents. It is also based on the assumption that the dummy agent does not pay for the resources used at the remote agent host. This seems unlikely, especially in the more general case where the target agent is a user agent and the task being delegated is more than just the execution of the agent (i.e. the delegation comes in the form of a service request from the dummy agent). Services and resources are not generally free and therefore in the event of the trust evaluator agent incurring costs – these would have to be passed on to the client agents.

After the dummy agent returns, the trust evaluator agent can analyse it. This constitutes a deliberative process whereby the trust evaluator agent determines if the dummy agent has succeeded and whether or not it experienced any interference. Interference could be detected if the agent’s executable binary code does not match the binary code of the dummy agent prior to migration. Similarly, it is possible to detect tampering of the data state. Also, it is possible to determine if the agent executed its task successfully or not. The dummy agent will either have computed some result (the correctness of which may be verifiable) or affected some change in the environment (detectable via perceptual input). Further discussion on detection mechanisms as solutions for ensuring execution integrity of mobile agents can be found in the literature [CLA03].

At this point it becomes obvious that this plan has its limitations. Detection mechanisms for ensuring execution integrity appear difficult to implement [CLA03].
Furthermore, it is exceptionally difficult (if not impossible) to determine if the agent’s privacy has been violated. A malicious host controller agent could make a copy of the mobile agent and attempt to disassemble the executable binary code. Attempts to read the data state seem exceptionally difficult to detect.

Nonetheless, the analysis of the dummy agent provides the necessary input into a rule-based reasoning process that allows the trust evaluator agent to attribute the success or failure of the delegated execution task. This is depicted in the final section of figure 10.2. It is at this final step that the agent determines (subjectively) if the success (respectively failure) is due to the target agent’s betrayal or due to some interference caused by an unreliable environment. Depending on the result of the reasoning process, the trust evaluator agent will revise its beliefs appropriately. This implies a change in the certainty of the agent’s beliefs.

The state of the environment does not change as a result of the execution of the plan. The agent however has gained further experience and revised its beliefs. The expectation is that incidents of betrayal will be minimised as a consequence. This should be reflected by the state description.

Sub-section 10.3.5 considers a plan to establish direct experience. At this point, it is clear that there are certain difficulties associated with the plan to establish direct experience. Constructing dummy agents suitable for arbitrary environments is probably not feasible. Moreover, analysing a dummy agent or its interactions with other agents seems a very difficult task. Therefore the plan is not considered further in the dissertation, nor implemented. Despite the apparent difficulties associated with the use of dummy agents for establishing direct experience, the idea still holds merit and could be investigated as part of future work. The next sub-section introduces another more practical plan with similar aims.

10.3.6 Eliciting Reputation Information

The trust evaluator agent may instead elect to invoke a plan to gather reputation information from appropriate sources. This approach represents a more feasible option when compared to establishing direct experience. However, direct experience in general is more reliable from the perspective of the trust evaluator agent concerned.
The preconditions for executing the plan to elicit reputation information are similar to the plan to establish direct experience. At least one host controller or user agent must be known to the trust evaluator agent. Furthermore, the trust evaluator agent must know of at least one available peer trust evaluator agent.
The trust evaluator agent should start by selecting target agents as depicted in the representation of the plan in figure 10.3. The trust evaluator agent needs to select another trust evaluator agent to carry out the evaluation. This can be done randomly or can be based on whichever trust evaluator agent is currently the most trustworthy for the requesting trust evaluator agent. A host controller agent or user agent must also be selected. The same options on which to base selection in the context of establishing direct experience are applicable here. It is also worth noting that the selection must also consider the task that has been or will be delegated. This is because trust is evaluated relative to a specific task.

The next step of the plan involves the trust evaluator agent taking the necessary action to send a message to the selected trust evaluator agent. The trust evaluator agent must request a detailed evaluation of the selected target agent. The idea is that the selected trust evaluator agent will reveal its beliefs relating to the target agent with respect to the selected task. Additionally, an ontological description indicating the global trust that the delegated task or goal will be achieved could be provided.

Typically the ontological term alone will be provided to non-trust-evaluator-agents (user or client agents). Therefore only evaluator agents will share their beliefs. Such a scheme however, is based on the assumption that all trust evaluator agents in the environment are able to share their beliefs. This means that any two trust evaluator agents must have equivalent beliefs (in terms of content), and must know that the content of their beliefs is equivalent. This is typically problematic if beliefs are shared between trust evaluator agents with different architectures, designs and implementations.

The final step of the plan to elicit reputation information involves a process of belief revision. The trust evaluator agent’s beliefs that relate to the target agent and the selected task could be updated based on the provided beliefs and the trust that the trust evaluator agent places in its selected peer.

Again, the plan to elicit reputation information results in a change in the agent’s beliefs and not a physical change in the agent’s environment. Similar to establishing
direct experience however, it should be the case that in the vast majority of situations incidents of betrayal are reduced.

Figure 10.3 A Plan for Eliciting Reputation Information

10.3.7 Discovering Relationships

Thus far two potential options for improving the trust evaluator agent’s evaluations have been considered. Relationship analysis is another option that would allow the agent to improve evaluations. Figure 10.4 provides an abstract example of a plan for relationship analysis.
Figure 10.4 A Plan for Relationship Analysis

The first step of the plan is to select a target agent for evaluation. A node representing the target agent must be added to the trust evaluator agent’s relationship graph. At the
same time a node representing the selected client agent must also be added to the relationship graph.

The next step of the plan presupposes that an ontology-based service directory exists. In other words, it has been assumed that the trust evaluator agent has access to an ontology-based ‘yellow pages’. The ‘yellow pages’ identifies the services provided by each agent (or its sphere of influence). The ontology defines what a particular service means.

Assuming the existence of an ontology-based ‘yellow pages’, the trust evaluator agent can firstly establish what service is offered by the target agent. Then for each other agent represented by a node in the graph, the trust evaluator agent similarly determines what service is provided. Once each agent has been characterised according to the service provided, the list of agents in the graph can be traversed and rules can be applied. These rules can be specified in the form of implications and determine the relationships between agents.

The final step in the plan must be executed separately if relationship analysis is conducted independently of the handling of a trust evaluation request. This step involves traversing the relationship graph and applying a different set of rules in order to revise beliefs. As depicted in figure 10.4, if a competitor relationship is established between a client agent and a target agent this should result in the target agent being less trustworthy for that particular client agent. This is an example of a pre-defined rule that could be used to reach improved evaluations of trustworthiness.

10.3.8 Detecting Betrayal
A part of handling a request for trust evaluation involves handling feedback and then detecting whether or not betrayal has occurred. Sub-section 10.3.8 describes an attribution plan to detect whether or not betrayal has occurred.

For the attribution plan to be executed it must be the case that the client agent has requested an evaluation of the same target agent, with respect to the same task on at least one distinct, previous occasion. Therefore it does not make sense to execute the attribution plan if the client agent is not previously known to the evaluator, nor does it
make sense to execute it if the requested evaluation is with respect to a different task or goal. Admittedly however, the plan could be adjusted to allow for the client’s prior opinion of the target agent to be conveyed to the trust evaluator agent even if the client agent is unknown and has never delegated a goal to the target agent. The trust evaluator agent could then take into account the client agent’s opinion and personality. For example, the client agent could convey a low opinion of the target agent if the client agent is risk averse.

Figure 10.5 provides a diagrammatic overview of the plan. As depicted, the client agent should provide feedback. The feedback provided will include the client agent’s perception of the outcome of the delegation (where delegation has taken place). Feedback should also include whether or not interference in the environment was perceived at the time of delegation. This may be easier to determine than a betrayal by the target agent. However, this does assume the existence of a log of events in the environment from which interference can be determined.

Once this feedback has been received, the trust evaluator agent can then attempt to attribute the success (respectively failure) of the delegated task or goal to environmental factors or agent internal factors. This will in turn result in a revision of the certainty of the trust evaluator agent’s beliefs of trust.

Honest feedback provided by client agents is clearly of value to the trust evaluator agent. Truthful and reliable feedback will result in improved evaluations. By requiring client agents to provide feedback (where applicable) for continued service, the trust evaluator agent gains access to potentially useful information. Thus the trust evaluator agent is able to provide a superior service. Therefore client agents will have incentive to provide feedback because the client agents are reliant on the service provided by the trust evaluator agent.

It is worth bearing in mind that if a rational client agent intends to use the services of a trust evaluator agent it is surely not in the agent’s interests to impair the trust evaluator agent in any way. Furthermore, since the trust evaluator agent continuously seeks reputation information the trust evaluator agent will quite possibly have
reputation reports from multiple agents. It is further possible that some of these reports may relate to the same agent and the same delegated task or goal.

![Diagram showing the process of detecting betrayal between an evaluator agent and a client agent.]

**Figure 10.5 Detecting Betrayal**

In such cases it becomes possible for the trust evaluator agent to determine if feedback received is inconsistent with the majority of reputation reports. Therefore, a deceitful client agent risks being found to be incompetent and thus untrustworthy in the event that it attempts to deceive a trust evaluator agent. Such reasoning of course, is based on the assumption that a truly malicious agent will try to betray multiple agents (not just the trust evaluator agent). As an added measure, based on the same assumption, the trust evaluator agent can also place less weight on that feedback. Thus the scheme is made more robust in the case of deceitful agents.
10.3.9 Evaluating Trustworthiness

Once feedback has been received and processed and other belief sources have been interrogated, the actual evaluation needs to be carried out. The plan outlined in figure 10.6 is a framework for the evaluation of trustworthiness.

**Figure 10.6 Evaluating Trustworthiness**

The plan represented in figure 10.6 is to be executed in response to a request for an evaluation. The agent also needs to be able to provide strengths (a measure of certainty) for the beliefs of trust used to reach an estimate of trustworthiness. In the case where the agent is performing the evaluation for the first time, the beliefs may be determined based on the agent’s ‘personality’. The agent might be designed to be pessimistic, cautious and risk-averse. This would imply that the agent is naturally suspicious. The strengths of these initial beliefs would then be adjusted by interrogating the appropriate belief sources as per the meta-plan.
Once a measure of trustworthiness has been computed the measure is mapped onto a set where the members of that set are described by an ontological term. This term is to be communicated to the client agent. If the client agent does not share the semantics, semantic mapping will need to be performed by the client agent.

The complete execution of the trust evaluation plan signals the end of the meta-plan to handle the request for reputation information. If no other requests have arrived in the interim, the agent is able to select other plans. The trust evaluation plan is at the centre of achieving the agent’s intention of minimising incidents of betrayal in the environment. Assuming the rationality and sound reasoning of the client agents it is expected that the trust evaluation plan will go some way to accomplishing this.

**10.3.10 Summary**

The trust evaluation plan may be at the centre of achieving the agent’s intention; however all the plans presented in section 10.3 play an important role. It should be clear that the plans for establishing direct experience, reputation, and relationships, all result in the revision of the agent’s beliefs. It is expected that a consolidated approach (i.e. the use of multiple plans) will result in superior evaluations. It also means that the agent will never be idle. The plan for servicing requests for evaluations ensures that the trust evaluator agent applies its beliefs and communicates its best judgement. The following section concludes this chapter.

**10.4 Conclusion**

Chapter 10 provides insight into the design of the trust evaluator agent. Firstly, desires and intentions are considered. The agent’s primary goal is to minimise incidents of betrayal in the environment. The possibility of decomposing the primary goal into further sub-goals was both discussed and rejected. The agent’s primary goal was also analysed in the same section. An overview of the available plans was provided at the start of section 10.3. The structure of plans as well as their selection was also addressed. Each plan is then decomposed into activities which were discussed.

On reflection, it is clear that to reach as complete and as sound as possible an estimation of another agent’s trustworthiness is a difficult and complicated process.
As mentioned in previous sections, many of the plans are expected to be time-consuming. Thus the need for trust evaluator agents is further justified. If the goal of minimising incidents of betrayal is not delegated to the trust evaluator agent, then it can be expected that the performance of other agents will be hindered. This is because either the agent will be unable to capitalise on time-limited opportunities (due to lengthy and complete evaluations), or it will be betrayed more frequently (due to shorter, incomplete evaluations). While the actual evaluation process may be time consuming in itself (irrespective of the agent that executes the plan), trust evaluator agents can use their ‘spare time’ to revise their beliefs through interrogating belief sources. This means that at the point of evaluation precious time is saved (at least as far as subsequent evaluations of the same agent with respect to the same task or goal are concerned).
Chapter 11
Embedding a Specialised Trust Evaluation Capability

11.1 Introduction
The trust evaluator agent’s specialised trust evaluation capability was presented as a series of dependant plans in the previous chapter. Those plans require certain deliberative capabilities that are conveniently implemented as a set of fine-grain components. The components collectively embed a complete, specialised, and deliberative trust evaluation capability into the trust evaluator agent. Furthermore, any user agent supporting the necessary interfaces can employ the relevant component(s) to gain a basic, but intelligent trust evaluation capability. This allows the user agents to evaluate the trustworthiness of the lower risk trust evaluator agents and then to delegate the goal of minimising incidents of betrayal in the environment.

Chapter 11 aims to provide an overview of the components with a conceptual level focus. Implementation details for components selected for implementation are deferred to chapter 12 on implementation.

Each section that follows outlines a potential component. This involves outlining the necessary interfaces, as well as detailing the capability to be provided by the component. More specifically, section 11.2 presents a conceptual overview of a component that embeds the trust evaluation capability. Section 11.3 introduces a component that will cater for the attribution of the outcome of a delegation. Section 11.4 proposes another component for attribution. Section 11.5 provides an overview of a component that provides the specialised relationship analysis capability. Section 11.6 proposes a component for handling reputation information.

11.2 A Component for Trust Evaluation
Section 11.2 introduces the most useful component. The component embeds the basic trust evaluation capability. The capability provided by the trust evaluation component is applied in the trust evaluator agent’s plan to evaluate trustworthiness. The trust evaluation component is also intended for use by other user agents. Figure 11.1
provides a diagrammatic representation of the component’s functionality and interfaces.

Figure 11.1 A Trust Evaluation Component

The trust evaluation component requires the specification of the strengths of various beliefs. The strengths of the beliefs should be derived from the trust evaluator agent’s subjective beliefs of trust as they apply to a specific target agent with respect to the potential delegation of a task $\alpha$ or a goal $g$. Hence the task or goal is implicitly specified through the strengths of the beliefs. It is worth emphasising that at no point – as far as this component is concerned – is it necessary to explicitly specify the semantics of the task or the goal.
Values between 0 and 1 must be specified indicating the certainty of the agent’s beliefs. An alternative although not recommended option is for the agent to specify various personality factors (pessimistic, risk-averse etc). Based on personality factors, default values for the strengths of beliefs can be applied. However, the use of personality factors and default values in particular is expected to lead to less accurate estimates of trustworthiness as it strays from the theory of evaluating trustworthiness relative to a task or goal.

The specified strengths of the beliefs are used in a computation (the sum of weighted products) that is derived from the topology of a belief network. One potential belief network is depicted in figure 11.2 below. The beliefs depicted stem from the trust theory by Castelfranchi et al, as discussed in chapter 3 [CAS01b]. Thus the belief network reflects knowledge of the beliefs of trust and the dependence relations between those beliefs.

![Figure 11.2 The Beliefs of Trust Network](image)

The component must clearly encapsulate an algorithm that embeds (at least implicitly) the knowledge of the beliefs of trust and the relations between them. As indicated, the algorithm will perform a simple computation by calculating the sum of the weighted products. The algorithm is easily extended to allow for feedback (a fraction of the final result is factored into the calculation) to cater for the influence of trust (see chapter 3). The algorithm has the advantage of not being overly complicated. The real challenge is establishing the most appropriate weights (an experimental process). Once this computation has been carried out, the strength of the trust that task $\alpha$ or goal $g$ will be completed or achieved successfully will have been computed.
The resulting value can then be mapped onto a set, the members of which are described by an ontological class. The process of mapping the resulting value to a set requires that some basic rules are also encapsulated by the component. Again, such rules are considered trivial. Determining the appropriate sets and concepts is more of a challenge.

The ontology used to define the concepts could be fairly trivial and should consist of only enough classes so as to avoid a distinction between negligibly different values. The resulting semantic term can then be returned to the participating agent.

Section 11.2 provides a conceptual overview of a component for evaluating trustworthiness. The trust evaluation component is a fundamental component that embeds the cognitive trust evaluation capability into a participating agent. The implementation of this component forms part of the next chapter. The next section however, focuses on another important component that embeds the capability for rational attribution of success (respectively failure) of a delegated task or goal.

11.3 A Component for Attribution

It is especially important for the participating agent to determine if it has been betrayed. Naturally if the agent is unable to determine this, then the agent is an ineffective evaluator and thus vulnerable due to its inability to avoid malicious entities. The attribution component also plays an important part in the use of direct experience (should it ever be used) as outlined in the previous chapter. The component is depicted in figure 11.3.

The attribution component requires the specification of three Boolean values. The values correspond to the answers for the three questions depicted in figure 11.3. The use of this component is based on the assumption that those questions can be answered.

Detecting environmental anomalies is not necessarily trivial but in many cases could be achieved by looking for particular events as recorded in log files. Determining agent interference is arguably much more difficult to achieve. The agent’s perception
of failure or success is another important input that is not necessarily trivial for the agent to determine. If evidence of failure or success happens to fall outside the agent’s viewable environment, then this could be difficult to impossible for the agent to determine.

Figure 11.3 A Component for Attribution

Given the Boolean values, a rule-based approach can be used to attribute the outcome of a delegation. The component simply embeds rules in the form of conditional statements. The consequents of these rules determine an increase (respectively decrease) in the agent’s trust in the target agent (respectively environment). Alternatively, a binary decision tree and associated algorithm could serve the same
purpose – as depicted in figures 11.3 and 11.4. With reference to figure 11.4, the consequents of the rules are now discussed in order of the external nodes (numbered 1 to 8).

![Figure 11.4 Attribution Decision Tree](image)

If the left most external node is reached then both the trust in the target agent and the environment should increase slightly. If node 2 is reached then both the trust in the target agent and the environment should decrease slightly. This results in an agent becoming suspicious of both the environment and the target agent. Furthermore, this could also indicate that the agent concerned is incompetent at detecting environmental anomalies or determining agent interference. Thus the scheme could be extended to ensure a decrease in the certainty of the agent’s self confidence belief – assuming an agent with such complex belief structures.

In the case that node 3 is reached, the trust in the target agent should decrease substantially since this indicates that betrayal has occurred. However, trust in the environment should increase substantially, if the task or goal was completed or accomplished successfully. This is perhaps an unlikely situation since in many environments the presence of opportunity and lack of threats alone is not enough to ensure the successful completion or achievement of a task or goal. Similarly, if node 4 is reached the trust in the agent decreases substantially while the trust in the environment should remain unchanged.

In the case that node 5 is reached, the trust in the agent should increase substantially because the agent was either able to overcome problems in the environment (thus demonstrating superior competence) or persisted to complete or achieve the delegated
task or goal (thus substantiating the disposition belief). Should node 6 be reached, trust in the environment should decrease substantially. Trust in the target agent should remain unchanged, although it could also be reduced slightly to reflect incompetence or lack of persistence in the presence of adversity caused by the environment.

In the unlikely event of node 7 being reached, there should be a substantial reduction in both the trust of the agent and that of the environment. It is unlikely that a task or goal will be successfully completed or accomplished given such betrayal. In the more trivial case that node 8 is reached then both the trust in the target agent and the trust in the environment should decrease.

The attribution component should return an appropriate instruction to increase (respectively decrease) trust in the environment (respectively target agent). The extent of the increase or decrease should also be specified.

Depending on how much information is available to the agent, this decision tree could get quite complicated. The component, as designed here, is based on the assumption that minimal information is available. However, if the trust evaluator agent could establish the reasons for an outcome then a more accurate decision tree could be built. For example, in the case that environmental anomalies are reported; no agent interference was detected; and success was perceived (as is the case if node 5 is reached), then it is useful to know the reasons why the target agent was successful. If evidence is available to suggest that it was due to superior competence then this could lead to an increase in the certainty of that particular belief.

The ‘bottom-up approach’ (individual modification of the certainty of individual beliefs) is better than the ‘top-down approach’ (uniform modification of the certainty of all beliefs) followed here. Simply indicating an increase (respectively decrease) in the trust in the target agent translates into a uniform increase (respectively decrease) in the certainty of all the beliefs of trust that relate to the trust in the target agent. A blanket increase or decrease is undesirable because the agent may only have

---

15 The bottom-up approach is only feasible when the trust evaluator agent has complete information (which is unlikely). Hence the top-down approach (which assumes minimal information is available) was followed.
succeeded because it persisted to do so – as opposed to superior competence. Therefore only the disposition belief should be affected. Such accurate fine-grain changes are not possible in the currently specified scheme. However, this is the trade-off that exists due to incomplete information that may be the reality in a multi-agent system (particularly open multi-agent systems, designed and developed by third parties).

The component also suffers from limited applicability; it can only be used in the case where there is certain evidence relating to the outcome of the prior delegation. The client agent must be able to perceive failure, agent betrayal, and environmental anomalies. The component presented in the following section attempts to provide an alternative.

**11.4 A Bayesian Attribution Component**

In contrast to the rule-based approach considered in the previous section, section 11.4 looks at an approach to attribution that attempts to cater for the inherent uncertainty in the case where the client agent is unable to directly perceive failure, agent betrayal and environmental anomalies.

The use of the Bayesian attribution component (illustrated in figure 11.5) assumes that the client agent is able to perceive at very least some of the following\(^\text{16}\): a malicious action; incompetent action; a lack of disposition; a lack of opportunity; or unhelpful environmental events.

The perception of some of the above mentioned incidents constitutes evidence that is required by the component. The component then uses the evidence in determining the conditional probability of environmental anomalies, agent betrayal, and less trivially, failure.

The component encapsulates a Bayesian network and approximate inference algorithm. The nodes in the Bayesian network correspond to Boolean random

---

\(^{16}\) The assumption is reasonable given that most trust models rely on the agent’s observations [ESF01]. The limitations of an agent’s ‘sphere of visibility’ introduce uncertainty. Therefore Bayesian networks are applicable to the problem of attribution [ESF01].
variables, each with its own associated conditional probability table. Thus, intuitively
the evidence must be Boolean values indicating whether or not an event or incident
occurred. The edges between the nodes represent causal knowledge. For example, an
edge between the node corresponding to incompetence and the node corresponding to
failure indicates that incompetence has the capacity to cause failure.

Once the conditional probabilities of agent betrayal, environmental anomalies and
failure have been determined, rules similar to those used in the attribution component
(section 11.3) can be applied. The rules result in belief revision instructions that are
returned by the component.

*Figure 11.5 A Bayesian Attribution Component*
The Bayesian attribution component also suffers from limited applicability in cases where the nature of the target agent’s actions cannot be perceived. Nonetheless, the component provides an alternative to the attribution component. Another advantage of the use of the component lies in the possibility for fine-grain belief revision due to the existence of perceived evidence. The use of a Bayesian network to implement fine-grain belief revision in the attribution component of section 11.3 was considered. Unfortunately, it was never intuitively clear how to structure such a network.

The previous two sections covered components for attribution. The following section presents a component for relationship analysis.

11.5 A Relationship Analysis Component

The relationship analysis component embeds a relationship analysis capability into the participating trust evaluator agent. The relationship analysis capability is intended to be used in conjunction with the trust evaluator agent’s plan for relationship analysis as discussed in the previous chapter. The proposed component embeds some of the procedural knowledge of the plan for relationship analysis and therefore simplifies it. The component is depicted in figure 11.6 below.

The use of the relationship analysis component assumes that there is a semantic description that describes agents in an application domain in terms of the services they provide. In particular it is assumed that the client agent and target agent are both adequately characterised by the classes in the application domain ontology.

The component therefore requires that the identity of both the client and target agents (their respective Uniform Resource Identifiers (URIs)) is provided along with the URI of the application domain ontology.

Given the URIs of the agents and the ontology it is possible to query the ontology to determine the agents’ characterisations. A comparison of the characterisations using rules allow for relationships to be identified.
In addition to the rules for identifying and establishing relationships, rules to determine changes in the strengths of beliefs based on the presence of relationships are included. These rules are simple implication rules (if-then rules). The antecedent should state the existence of an identified relationship. The consequent should indicate the extent of the increase or decrease in a particular belief of trust relating to the target agent.

As an optimisation, nodes representing the specified agents need to be added to the relationship graph if such nodes do not already exist. Then for each newly added agent, it is necessary to identify relationships with the other agents already
represented in the graph - where those relationships exist. This is to be achieved by comparing the characterisations of the agents, as described above. The identification of relationships determines if new edges need to be added to the graph. New edges indicative of relationships are added to the graph where necessary.

Once the relationship(s) between the client agent and the target agent have been established, the appropriate belief revision rules can be applied. Application of these belief revision rules results in an instruction to increase or decrease the strengths of one or more beliefs of trust relating to the target agent. For example, if the competitor relationship has been established, it is rational to place less trust in the target agent. Therefore, the strength of the disposition belief should be reduced because it is less likely that the client agent’s competitor will persist to help the client agent. Similarly, the client agent may reason that the competitor relationship makes the target agent more likely to betray it. Therefore, the safety belief may also be reduced. However, caution does need to be exercised in increasing or decreasing the strengths of individual beliefs of trust. It must be remembered that trust is evaluated with respect to a particular task or goal. Therefore, care needs to be taken in the design of the rules; for example the existence of the competitor relationship is unlikely to affect the trust evaluator agent’s competence belief.

The instruction to increase (respectively decrease) the strength of one or more beliefs and the extent of the increases (respectively decreases) is provided by the component. The belief revision instructions can then be used in adjusting the trust evaluator agent’s beliefs of trust immediately prior to evaluation.

This section introduces a specialised component, intended primarily for use by trust evaluator agents. The component embeds a relationship analysis capability into trust evaluator agents. It is expected that this capability will allow these agents to reach more accurate estimates of other agents’ trustworthiness. The following section looks at one final component.
11.6 A Reputation Processing Component

A final component to provide a complete trust evaluation capability is the reputation processing component. A sensible design choice for such a component requires three inputs. The first input is the strengths of the beliefs of trust as provided by the reputation provider. The second input is a credibility value indicating the level of trust that this agent places in the reputation information and the reputation provider. More advanced implementations could decompose the single credibility value into a more complex set of values. The third required input is the trust evaluator agent’s own beliefs of trust.

The component embeds the functionality to calculate an influence value (a weighted product using the reputation provider’s beliefs of trust and the credibility value) for each belief of trust. A comparison of the reputation provider’s beliefs and the trust evaluator agent’s beliefs is required to determine whether the influence is positive or negative. The computed influence values constitute belief revision instructions that are returned by the component. The trust evaluator agent can then revise its beliefs as appropriate.

More complex implementations of a reputation processing component are certainly possible. However, the extent of an increase in the utility offered by a more complex implementation is questionable.

11.7 Conclusion

Chapter 11 introduces five components that embed intelligent trust evaluation capabilities into participating agents. The components are all intended to be embedded into trust evaluator agents in support of the relevant plans as outlined in the previous chapter. However, the component for trust evaluation (section 11.2) and the component for attribution (section 11.3) could also be used to embed a very necessary capability into any compatible social agent in a multi-agent system or agent society.

The component-based approach demonstrated here provides a convenient way to encapsulate intelligent and reusable functionality. The well established benefits of the component-based approach also apply; as new, more complex techniques aimed at
trust evaluation are developed, the components can be modified to endow the agent with improved capability – without having to redevelop the entire agent program. As with any component-based approach, difficulties will arise with changes in interface design however.

Perhaps the most salient benefit of the component-based approach is that it shields the agent developer from the complexity of the problem at hand. It eliminates the requirement for the agent developer to have knowledge of trust theoretic approaches and techniques, thus making the developer’s job easier.

Chapter 11 concludes a conceptual level specification of a model aimed at minimising incidents of betrayal in the agent environment, through the identification and subsequent isolation of malicious agents. The chapter that follows details the design and implementation of an exploratory prototype that realises selected functionality.
Chapter 12

Design and Implementation

12.1 Introduction

Chapter 12 outlines the design and implementation details of a prototype implementation of the reputation-providing trust evaluator agent (hereafter referred to as REVALUATOR) outlined in the previous four chapters (chapters 8-11). The work in chapter 12 differs from the work in the previous four chapters; those chapters deal with a conceptual model, its architecture, and emergent characteristics. The work in chapter 12 deals with the more practical design and implementation details of REVALUATOR - albeit at a high level. The full lower-level implementation details are highly complex, protracted, and deemed not particularly relevant for a sufficient understanding of the implementation of REVALUATOR. As such, the full lower-level implementation details are relegated to the appendices.

The model and its applicability are put to the test through the implementation of REVALUATOR. The REVALUATOR prototype also serves as a means through which results can be derived. The results and analysis thereof are the topic of the next chapter.

This chapter is structured as follows: A brief overview of the approach followed is given in section 12.2. Design details relating to the structure of the software are provided in sections 12.3 and 12.4. A brief discussion on the implementation details is included in section 12.5. An overview of REVALUATOR, including screenshots are provided in section 12.6. Section 12.7 provides a brief analysis of the design and implementation in concluding the chapter.

12.2 Approach

Section 12.2 provides some insight into the design and development of REVALUATOR by briefly describing the approach followed. The approach can best be described as agile, iterative, incremental, and exploratory. The first phase of design involved basic use case analysis and the specification of a conceptual model. The
conceptual model provides abstractions necessary to handle the complexity of the problem. The conceptual model forms the basis of the previous four chapters. The conceptual model served as a guide for a further design phase (an object-oriented design phase), during which a preliminary class model was developed. The class model is aimed at realising the architectural abstractions of the conceptual model. The produced class model is also the basis for the design work presented in this chapter.

Subsequently, exploratory work began on the implementation of REVALUATOR. The abstractions comprising the agent (the infrastructure) were developed first. Afterwards, work began on the components. The components were developed sequentially. Component development involved developing unit tests for the component, followed by developing the component in a bottom-up fashion (meaning relevant data structures were developed and tested first).

The exploratory development revealed the need for new classes and changes to interfaces. The development also resulted in minor changes to the conceptual model. These changes filtered down to the class model and ultimately the software itself through further iterations of the process described above.

Although this process resulted in a working prototype, certain inadequacies can be identified. These inadequacies receive attention in the concluding chapter of this dissertation. The end result of the described process is presented in the remainder of this chapter.

12.3 Specialised Trust Evaluator Agent Design

12.3.1 Introduction

The architectural abstractions presented in the previously defined trust evaluator agent architecture of chapter 9, manifest themselves in the form of Java classes. These classes provide the trust evaluator agent infrastructure needed in order to embed an intelligent trust evaluation capability. The trust evaluator agent is actually an extension of a standard Java Agent Development Framework (JADE) agent, thus the classes developed as part of REVALUATOR can be seen as sitting on top of the JADE agent infrastructure as depicted in figure 12.1.
A discussion of the JADE agent framework and the JADE agent architecture falls outside the scope of this work. It suffices to say that the underlying JADE agent infrastructure provides the ideal support foundation. The most noteworthy support includes the Foundation for Intelligent Physical Agents (FIPA) agent communication language (ACL) compliant agent communication, intra-platform mobility, and agent life-cycle management.

![Figure 12.1 Contextual Model: Architectural Relationship showing the Specialised Trust Evaluator Agent as a JADE Agent](image)

The remainder of this section deals with the classes (and relations between them) that comprise the trust evaluator agent in accordance with the architectural model specified in chapter 9.
12.3.2 Specialised Trust Evaluator Agent Class Model

The trust evaluator agent is a deliberative agent exhibiting a specialised BDI-theoretic architecture. The architectural abstractions specified by the architectural model correspond to classes in the implementation, given that an object-oriented development paradigm is most practical. The static structure of the agent (the classes comprising the abstractions, their relationships and interactions) is expressible using Unified Modelling Language (UML) class models.

However, certain difficulties arise when trying to provide a complete representation of the static structure of the agent at the implementation level using UML class modelling. The number of complex composite classes and the high coupling between them mean that the models become very large, with numerous relationships/interactions (even with package views partitioning model elements). Apart from the difficulty of actually displaying models of that size on paper, there is little utility in displaying the verbose and overcomplicated models. The implementation level models are only relevant for a prospective developer of REVALUATOR (even then, they are arguably overwhelming). It is therefore more useful to provide a slightly higher, conceptual level view of the implementation of REVALUATOR (this view is still at a much lower level than that depicted by the architectural model of previous chapters).

Thus, the subsequent models of this section omit unnecessary complexity. The models only reflect the existence of the relevant classes, and the fundamentally important relationships and interactions between those relevant classes. Additionally, the package models of the static structure of the agent do not lend themselves well to an explanation of the design of the agent. As such, the presented models deviate from the package models\textsuperscript{17}.

The UML use case diagram of figure 12.2 provides a contextual framework for discussing the models. The static structure of the agent is revealed in figures 12.3 – 12.7. The structure of each component is presented afterwards.

\textsuperscript{17} Full static structure models indicating the relevant packages are provided in appendix A.
Figure 12.2 Specialised Trust Evaluator Agent Use Case

The use case indicates that client agents will interact with the trust evaluator agent in order to make use of the trust evaluation service being provided. This means that the client agent will send a message to the trust evaluator agent. The trust evaluator agent must perceive the percept (the message) and then interpret it. As a result of the agent’s interpretation of the percept, the agent's knowledge of the state of the environment (hereafter referred to as state description) is updated. Much of the underlying communication details are taken care of courtesy of JADE. JADE provides FIPA-compliant agent communication support. With the underlying communication details safely delegated, the developer can focus on interpretation. The internal structure of the agent that facilitates the perception and interpretation of percepts is shown in figure 12.3.

The updating of the agent’s state description allows the agent to choose a course of action partly based on its ‘current knowledge of the world’. The process of choosing a course of action is heavily dependant on the agent’s state description. The state description may define an opportunity for achieving a goal. In this case, the goal is
actually achievable given the opportunity and is therefore a candidate for selection from the agent’s goal library (hereafter referred to as desires). The selection is made possible by defining a state description for each potential goal a priori. Ideally the state descriptions constituting opportunity requirements should be specified using Extensible Mark-up Language (XML) for maximum flexibility and maintainability.

Figure 12.3 Interpretation of Percepts

Once a goal is selected, that goal becomes the agent’s intention – meaning the agent intends to achieve it. At this point the agent must select a plan for achieving the intention. Several possibilities exist as far as plan selection is concerned; the simplest of which is to randomly select a plan that is known in advance to be suitable for achieving the intention. Another possibility is to compare the current state description to the plan’s precondition (a state description specifying the conditions that must prevail for the plan to be executed successfully). Since the trust evaluator agent has
only one goal and therefore one intention, a comparison of state descriptions is all that
is required. Since a deterministic approach is favoured, the latter approach to plan
selection was adopted. The model of figure 12.4 illustrates the classes used to
implement plan selection.

Once a plan has been selected the various activities that comprise that plan need to be
executed. At the simplest level a plan can be viewed as an object encapsulating a
precondition, optionally a post-condition, and a body. The body of the plan can be
seen as a sequence of numbers, where each number identifies an activity to be
executed. If the activity is a deliberative activity then the identifier needs to be passed
to the deliberation library. The deliberation library will use the identifier to identify
which deliberative process needs to be executed. The deliberation library then
bootstraps the relevant deliberative process. The deliberative processes should
typically only rely on the agent’s knowledge and/or beliefs as input. The deliberative
processes should then result in the update of the agent’s knowledge and/or beliefs.
The process of bootstrapping deliberative processes is illustrated by the model of
figure 12.5.

In the context of REVALUATOR the knowledge mentioned above corresponds to the
agent’s state description. The beliefs correspond to the beliefs of trust with respect to
a particular agent with respect to the achievement of a particular goal.

As can be seen in figure 12.5 the agent’s belief base is comprised of belief sets. There
will be a single belief set for the agent’s environment and a single belief set for each
distinct target agent and for each distinct goal that may be delegated to that target
agent. Each belief set is comprised of distinct beliefs. These beliefs are simply the
beliefs of trust. The beliefs of trust in an agent differ from the beliefs of trust in the
environment hence the distinction between an environmental belief set and an agent
belief set. It is possible however, to simplify the design by modelling the environment
as an agent.
Figure 12.4 Plan Selection
As a consequence of having distinct beliefs, each time a new target agent becomes known (via evaluation requests from a client agent) a new belief set will have to be instantiated. Furthermore, even if the target agent is known to the trust evaluator agent, the trust evaluator agent’s beliefs relate to a specific goal to be delegated. Therefore, if an evaluation request is received for a known target agent, but with respect to an unknown or new goal, a new belief set will be required.
Deliberative activities are not the only type of activity that can be specified in the body of a plan. The activity identifiers that are encapsulated by a plan object may also refer to actions. In such cases, the controller passes the identifier to the actuator (that also serves as a library of actions).

The actuator identifies which action to execute and then executes the action. Actions in the context of the trust evaluator agent refer to an atomic act that causes a change or event in the environment. Given the problem domain, the actions are almost certainly guaranteed to be communicative acts directed at client agents or other trust evaluator agents.

In the event of some communicative acts the agent may need to communicate a result (information that can be retrieved from the state description), or communicate a request (in which case the state description must be updated).

A separate class defines each distinct action. Therefore, if a new action were to be added then a new class would have to be defined. The set of actions are thought to be finite and static. As such it is not expected that new actions will be added. The classes involved in executing actions are illustrated in the model of figure 12.6.

As is evident from the class models of figures 12.3 – 12.6 the trust evaluator agent’s state description plays an important and central role. In fact, the importance and centralisation of the state description was not anticipated at the outset. It was only in later iterations of design that the need for a comprehensive and more complex state description became evident. The model of figure 12.7 illustrates the classes that comprise an ideal state description.

As can be seen in figure 12.7 the state description is made up of three types of state. Conversation state reflects the state of the trust evaluator agent’s conversations with other agents. In this way, the trust evaluator agent is aware that it has for example made a request and is awaiting a response. The process state reflects the state of the agent’s deliberations. The process state allows the agent to keep track of what deliberative processes have been executed in the pursuit of the complete execution of the meta-plan for trust evaluation. The environmental state allows the agent to keep
Figure 12.6 Executing Actions
track of the state of environmental variables (for example, whether or not environmental anomalies or agent betrayal has occurred). The state classes of figure 12.7 define mostly Boolean variables the values of which are to be encapsulated by instantiated state objects in the running agent program.

Section 12.3.2 provides an overview of the static structure of the trust evaluator agent. The structures used in the interpretation of percepts; the selection of plans; the bootstrapping of deliberative processes; and the execution of actions have been illustrated in the various models. Additionally, attention has been paid to the structures involved in catering for the agent’s management of state. The following section builds on this overview by providing a high-level indication of the runtime interactions between objects.
12.3.3 Specialised Trust Evaluator Agent Controller Interaction Model

Arguably the best way to cement an understanding of the internal workings of the trust evaluator agent is with the aid of a UML interaction-sequence diagram. Figure 12.8 below provides an interaction-sequence diagram indicating the runtime interactions (at a high-level) between many of the classes from the previous models.

Figure 12.8 shows the controller invoking the sensing and interpretation of percepts; the selection of a plan; and the execution of the plan. The diagram shows the interactions in the case where the plan consists of an arbitrary deliberative activity. The interactions follow much the same pattern in the case where the plan consists of an arbitrary action, or sequence of actions.

The algorithm for the controller was adapted from the various controller algorithms appearing in the book by Wooldridge [WOO00]. The particular algorithm concerned executes the plan to completion (without sensing and interpreting percepts during execution). While this results in a simpler algorithm, difficulties are introduced where responses to communicative acts are required in order to continue the evaluation. One solution to the problem is to decompose plans into simpler plans. At the end of the execution of the plan the controller invokes the sensor and interpreter and subsequently the next plan can be selected and executed. REVALUATOR has been implemented in this way because the main focus is on proving the concepts of the model and determining the extent of the limitations of the trust-based approach. There could be more elegant solutions, but implementation of more elegant solutions invariably detracls from the main focus. One such elegant solution is to adopt a more complex controller algorithm. Another arguably more elegant possibility is to treat perception as an action and to include the perception action in plans as required.

12.3.4 Summary

This section presents the fundamental design of the trust evaluator agent. The main focus of the section is the static structure of the agent as presented in the models of figures 12.3 – 12.7. The structures involved in perception and interpretation; plan selection; bootstrapping deliberative processes; and executing actions have been discussed. Attention has also been paid to the structures that comprise the agent’s
Figure 12.8 Controller Interaction-Sequence Diagram
beliefs and knowledge. The section that follows looks at the structures involved in providing the agent’s deliberative capabilities.

12.4 Deliberative Component Micro-Design

12.4.1 Introduction

The trust evaluator agent’s deliberative capabilities (its trust evaluation capabilities) are provided by the various components discussed previously in the dissertation. The invocation of the facilities provided by the components is the responsibility of the trust evaluator agent’s deliberation library object. The invocation or bootstrapping of deliberative processes is indicated in the class model of figure 12.5. This particular section presents the design of these components. Each component consists of classes, the most important of which are indicated in the static structure models of this section.

Section 12.4.2 details the design of the trust evaluation component. Section 12.4.3 presents the design of the reputation processing component. The relationship analysis component is the focus of section 12.4.4. The attribution component is presented in section 12.4.5, while the Bayesian attribution component is presented in section 12.4.6.

12.4.2 Trust Evaluation Component Design

The trust evaluation component is a simple component consisting of one fundamental class. The class provides an interface with one publicly accessible method. The method requires the beliefs of trust specified as floating point numbers in the range 0 to 1 inclusive. The method returns an integer value indicating the degree or level of trust. The values range from 0 (indicating low trust), to 4 (indicating high trust). The agent using the component must use these values to select an ontological term describing the level of trust. The ontological term may then be communicated to other agents. The static structure of the component is illustrated in figure 12.9.

12.4.3 Reputation Processing Component

The reputation processing component is illustrated in figure 12.10. The ReputationComponent class offers a single publicly accessible method. It requires two arrays of floating point numbers indicating the beliefs of trust. The first array
Figure 12.9 Trust Evaluation Component

```
TrustEvaluationComponent
```

Figure 12.10 Reputation Processing Component

```
BeliefRevisionInstruction
-beliefIdentifier : String
-modifier : float
+getIdentifier() : String
+getModifier() : float

ReputationComponent
+LOW_TRUST : int = 1
+MEDIUM_LOW_TRUST : int = 2
+MEDIUM_TRUST : int = 3
+MEDIUM_HIGH_TRUST : int = 4
+HIGH_TRUST : int = 5
+processReputation(in reputation : float[], in beliefs : float[], in credibility : int) : BeliefRevisionInstruction[]
```
corresponds to the beliefs of the reputation provider. The second array corresponds to the beliefs of the agent that requested the reputation. An integer indicating the level of trust that is placed in the reputation provider is also required. The method returns an array of BeliefRevisionInstruction objects that the agent can use to adjust its beliefs.

### 12.4.4 Relationship Analysis Component

The relationship analysis component is a more complicated component. The relationship analysis component relies on the Jena Semantic Web Framework API for inferring relationships from an application domain ontology [HP08]. Inferring relationships using an ontology and Jena seems to be a time consuming process. With this in mind, a graph data structure was included to allow for the ‘caching’ of discovered relationships among agents. Although the ‘caching’ was not implemented, it is thought that the caching scheme will reduce the time required to determine the existence of relationships in subsequent analysis.

The RelationshipAnalysisComponent class provides an analyse method that takes the Uniform Resource Identifier (URI) of the client agent and the URI of the target agent as arguments. The URIs are used to identify the agents in the ontology. The method returns an array of BeliefRevisionInstruction objects that can be used to update the agent’s beliefs. Currently, the component has only been designed to provide belief revision instructions on the basis of the existence of competitor relationships. It is possible to extend the scheme to include modification on the basis of the existence of other relationships.

### 12.4.5 Attribution Component

The attribution component relies on a binary tree data structure for embedding rules that can be used in deciding the extent of the revision of the beliefs of trust. The AttributionComponent class defines a single publicly accessible method that takes three arguments and returns an array of BeliefRevisionInstruction objects (figure 12.11). The arguments required include Boolean values indicating whether the client agent has been betrayed; whether the delegated goal was achieved; and whether environmental anomalies occurred.
Figure 12.11 Attribution Component
12.4.6 Bayesian Attribution Component

The Bayesian attribution component is another complex component. The component relies on a directed graph data structure to implement a Bayesian network (consisting of Bayesian nodes, where each node has a conditional probability table). The Bayesian network is encapsulated by the BayesianAttributionComponent class (figure 12.12). The Monte Carlo Markov Chain algorithm is used for approximate inference [RUS03]. Therefore, given sufficient evidence, it is possible to determine the conditional probability of a given query variable.

The BayesianAttributionComponent class exposes a publicly accessible method that takes an integer value identifying a query variable as an argument, along with evidence values. The method returns the conditional probability of the query variable.

The component is only useful in cases where the client agent has sufficient evidence. For example, if the client agent can establish for certain that the target agent has behaved in a malicious or incompetent manner then the conditional probability of betrayal by the target agent can be established. However, the problem is one of perception: the client agent may not be able to perceive the nature of the target agent’s actions. Furthermore, there is likely to be uncertainty in the interpretation of the perception of the target agent’s actions and their effect on the environment.

12.4.7 Summary

Section 12.4 provides a brief overview of the design of each component. All the components were implemented as JavaBean components – or beans. The JavaBean component model allows for portable and platform-independent components. The trust evaluation component and reputation processing component are the least complicated of the components. Conceptually, both components can be seen as embedding rules into the participating agent. Similarly, the attribution component can be seen as embedding this form of intelligence (rules). The relationship analysis component provides a more advanced capability through the use of an application domain ontology. Arguably, the most complex component is the Bayesian attribution component. The component strays from the rule-based approach followed in the design of the other components by relying on a probabilistic approach.
Figure 12.12 Bayesian Attribution Component
12.5 A Note on Implementation

Section 12.3 and 12.4 provide details relating to the design of the trust evaluator agent and the components respectively. Section 12.5 provides some insight into the implementation of the software.

As already mentioned, the Java programming language was used to develop the software (data structures, a client agent, a trust evaluator agent and associated components). Given the choice of programming language it made sense to implement the components as JavaBeans. Figure 12.13 shows a summary of ‘bean information’ for the relationship analysis component. All the components described in section 12.4 were implemented and tested using the JUnit unit testing framework. With one exception, all the implemented components are used by the trust evaluator agent. It was decided to use the rule-based attribution component instead of the Bayesian attribution component for attribution. The decision was taken not to use the Bayesian attribution component, primarily because it typically requires more information to be communicated between the trust evaluator agent and the client agent. Other anticipated difficulties included the problem of perceiving and interpreting the target agent’s actions (as malicious, incompetent etc.). Admittedly however, the problem of perception and interpretation was averted due to the client agent simulating all interaction and delegation with the target agent.

The choice of programming language was heavily influenced by Java’s portability and platform-independence benefits – particularly given the focus on mobile agency throughout this dissertation. The trust evaluator agent was implemented to run on the JADE platform as discussed in section 12.3. As indicated in that section, the communication support offered by JADE was leveraged to allow for communication between the trust evaluator agent and any other agent using the FIPA-SL0 language and the same domain ontology.

Ontology development was facilitated through the use of the Protégé Ontology Editor [STA08]. A frame-based ontology for generic communication with the trust evaluator agent was produced using Protégé as the screenshot of figures 12.14 indicates.¹⁸

¹⁸ See appendix B for all ontology models
Figure 12.13 Component Description Interface
Figure 12.14 Development of a Generic Trust Communication Ontology
A frame-based ontology was required (as opposed to an Ontology Web Language (OWL) ontology [DAC03, W3C07]) in order to produce the Java classes required by JADE. The Java classes were generated from the ontology using the Protégé Bean Generator plugin [VAN04]. The results of the generation can be seen in the screenshot of figure 12.15.

Protégé was also used to develop an OWL ontology for a telemanufacturing application domain [PIK08]19. This OWL ontology is used by the relationship analysis component to determine relationships among agents (instances in the ontology). Screenshots showing the classes, properties and design time instances of the telemanufacturing service ontology can be found in figures 12.16 – 12.18. The interrogation of the OWL ontology was made possible through the use of the Jena Semantic Web Framework API.

The section following the screenshots aims to provide an overview of the runtime functioning of REVALUATOR.

12.6 A Working Prototype: REVALUATOR

Section 12.6 presents an overview of REVALUATOR. The work presented in this section focuses on a general use case where a known client agent requests reputation information from a trust evaluator agent. The reputation information requested relates to a known target agent and a known goal.

At start-up the trust evaluator agent registers with the directory facilitator agent. The registration of the trust evaluator agent can be seen via the directory facilitator agent’s user interface (figure 12.19). A description of the trust evaluator agent must also be added to the application domain ontology. Similarly, descriptions for all other agents must be added to the ontology. For the purposes of REVALUATOR, the descriptions were added a priori using Protégé.

19 See appendix B for all ontology models
Figure 12.15 Ontology Bean Generator
Figure 12.16 Development of the Telemanufacturing Application Domain Ontology (Classes)
Figure 12.17 Development of the Telemanufacturing Application Domain Ontology (Properties)
Figure 12.18 Development of the Telemanufacturing Application Domain Ontology (Instances)
Figure 12.19 Registration of the Specialised Trust Evaluator Agent with the Directory Facilitator Agent
The client agent can then use the services provided by the directory facilitator agent to find trust evaluator agents. For the purposes of REVALUATOR there is only one trust evaluator agent, so selection of the trust evaluator agent is trivial.

For simplicity sake the client agent simulates all interaction with the target agent. In particular the client agent simulates the delegation of the goal and the outcome (i.e. whether or not the goal is achieved successfully).

The client agent may then request reputation information from the trust evaluator agent (for the purpose of reaching a hypothetical decision relating to the future delegation of the goal). Figure 12.20 shows a series of interactions between the trust evaluator agent and the client agent. The interaction highlighted by the figure shows the client’s trust evaluation request as it was sent to the trust evaluator agent.

After the trust evaluation request has been received by the trust evaluator agent, the trust evaluation meta-plan can be executed. Since the client agent, target agent, and goal are fixed for the purposes of REVALUATOR, no new belief set has to be instantiated. Instead, the trust evaluator agent may proceed immediately with the execution of the meta-plan. The first step involved when dealing with a known client agent, target agent and goal requires the elicitation of information relating to past direct experience. The client agent cannot make use of the trust evaluation service unless it supplies this information when it is available. Therefore there is some incentive for the client agent to provide reputation information (in a raw form).

In order to elicit the information in question, the trust evaluator agent will ask the client agent three questions. The answers to the questions allow the trust evaluator agent to attribute the success or failure of the previously delegated goal to agent internal factors or environmental factors. The trust evaluator agent demands to know if the client perceived agent betrayal; if the client perceived environmental anomalies during the execution of the delegated goal; and if the goal was accomplished. The agent betrayal request can be seen in figure 12.21. Figure 12.22 shows the three requests in the client’s ‘inbox’ before they have been processed.
Figure 12.20 A Trust Evaluation Request Message
Figure 12.21 An Agent Betrayal Request Message
Figure 12.22 The Client’s Inbox with Requests for Information useful for Attribution
As mentioned, the client randomly generates the responses to these questions. The client’s simulation of the target agent can be changed to simulate certain behaviour – a requirement for generating the results presented in chapter 13.

The client agent simply answers the questions using Boolean values, as is shown in figure 12.23. Once the trust evaluator agent has received the responses, it can then perform the attribution which results in belief revision. It should be pointed out that in theory more questions could be asked in an evidence gathering process and that the Bayesian attribution component could then be used instead.

At this point the trust evaluator agent may then decide to gather reputation information from other trust evaluator agents. To avoid the burden and complexity associated with agent communication the interaction with other trust evaluator agents was simulated. The results of that simulation were used as input to the reputation processing component. Reputation processing also results in belief revision.

The trust evaluator agent may also decide to establish the existence of any relevant relationships between the client and the target. Relationship analysis does not result in permanent belief revision. For example, if a competitor relationship exists, the beliefs used for the imminent evaluation are revised. However, the agent’s perennial beliefs are not revised. This is partly because relationships may change over time. Also, continually penalising a target agent by negatively revising perennial beliefs based on the current existence of a competitor relationship does not lead to reliable or accurate estimates.

After the above steps of attribution, reputation processing, and relationship analysis have been completed, the trust evaluator agent is then able to evaluate the trustworthiness of the target agent. The evaluation process results in a conceptual result that needs to be communicated to the client agent. The screenshots of figures 12.24 and 12.25 show the communicated results for two different evaluations. Figure 12.24 shows that the ontological concept of medium low trust is communicated to the client agent. For another evaluation the evaluator instead communicated the ontological concept of low trust for the same agent (figure 12.25).
Figure 12.23 The Client’s Response to the Specialised Trust Evaluator Agent’s Request
Figure 12.24 The Specialised Trust Evaluator Agent’s Response to an Evaluation Request (A)
Figure 12.25 The Specialised Trust Evaluator Agent’s Response to an Evaluation Request (B)
At this point the client agent must use the communicated result (amongst other factors) to decide whether or not to delegate the goal. However, this is not the focus of the REVALUATOR and as such was omitted. The client agent always delegates the goal to establish direct experience.

Section 12.6 provides some insight into the handling of a general case where a known client agent requests an evaluation of a known target agent in relation to the delegation of a default goal. The following section concludes this chapter.

12.7 Conclusion
The following paragraphs of the conclusion provide a holistic analysis of the approach, design and development. The main contributions of chapter 12 are briefly summarised in the last paragraph.

The approach used has numerous advantages. The agile, iterative, and exploratory nature of development was necessary given the uncertainty involved in design and development. The initial focus on architectural abstractions (i.e. the model chapters) was beneficial in handling the internal complexity of the trust evaluator agent. The partitioning of core functionalities into separate components also resulted in a separation of concerns that made development easier.

The inherent service orientation of the solution also provides an abstraction that would make a more complete multi-agent system development easier to implement. The same service orientation also allows for improved handling of heterogeneity in large, potentially open, multi-agent systems, especially given the use of ontology-backed FIPA compliant agent communications.

Several weaknesses can be identified. Modelling in particular was problematic. Given the internal complexity of the agent, producing low-level readable static-structure models was extremely difficult to the point of infeasibility. The approach to development failed to place enough emphasis on the complexity and dynamics of agent communication until fairly late in proceedings. More attention should have been paid to the agent’s communicative actions earlier in the process. Similar problems
were experienced with respect to the agent’s internal handling and representation of state. An agent-oriented development methodology paying specific attention to such matters may have been beneficial (if such a methodology exists).

Despite any weaknesses in the software engineering approach (which is not the main focus of the research), a realistic design that is consistent with the architectural abstractions has been implemented. REVALUATOR is fit for the purposes of demonstrating how an agent determines if other entities are trustworthy and therefore serves as suitable proof of concept as far as techniques for trust evaluation are concerned. REVALUATOR is also fit for the purpose of deriving results that can be used to determine the extent of the limitations of a trust-based approach.

REVALUATOR, despite being fit for purpose, could benefit from further implementation and fine-tuning. The instantiation and collaboration of multiple trust evaluator agents in a so-called web of trust would be an obvious improvement to be considered as part of future work.

Chapter 12 provides insight into the design and development of REVALUATOR. Specific attention has been given to the approach used during development, the design of the agent and associated components as well as to the implementation of the agent and associated components. The penultimate section attempts to illustrate the workings of the product of the design and development by considering the execution of a general use case. The chapter that follows analyses the results of the research as a whole (including the model chapters). The REVALUATOR prototype presented in chapter 12 was used to produce the results that appear in chapter 13.
Chapter 13  
Results  

13.1 Introduction  
The solution presented in the dissertation results in a specialised architecture for a cognitive trust evaluator agent. The trust evaluator agent can be added to any mobile agent system to provide a trust evaluation service. The implementation of a trust evaluator agent exhibiting the architecture, along with the associated components are (as outputs of research) results that are worthy of analysis. Since the design and implementation (chapter 12) verifies the applicability of the approach, the focus of the analysis must be on the efficacy of the approach/solution. 

It is expected that the solution is effective since the principles and techniques on which it is based are provably sound according to the literature. The results presented in chapter 13 are expected to confirm the value of those contributions and in particular the value of a consolidated approach to trust evaluation. 

Chapter 13 is arranged as follows: section 13.2 motivates some criteria for qualitative analysis of the implemented solution. Section 13.3 analyses the implemented solution using the established criteria as a basis. Section 13.4 establishes metrics necessary for conducting experiments. Section 13.5 presents the quantitative results of the experiments produced using the implemented solution. In section 13.6 these results are analysed. Section 13.7 concludes the chapter. 

13.2 Criteria for the Analysis of REVALUATOR  
In chapter 4 of the dissertation, a number of methods/models for trust evaluation were reviewed. These methods/models were subject to analysis according to the criteria set out in section 4.6. Similarly, the implemented solution shall also be subject to analysis according to the criteria specified and motivated in section 4.7.3.
In addition to the criteria specified in section 4.7.3 (which are listed below for convenience) the following criteria are also deemed relevant:

- Robustness.
- Reliability.
- Adaptability.
- Efficiency.
- Reusability (including generality, flexibility and scalability).
- Intelligence.
- Degree of autonomy.
- Security.

Since the derivation, discussion, and motivation of the first five criteria has been presented elsewhere in the dissertation the focus is on the last three criteria.

The first new criterion introduced here is intelligence. Since the research (in a broad sense) deals with agent technology as a specialisation of artificial intelligence, it is worth considering the degree of intelligence of the implemented solution. It is also important to establish the degree of intelligence provided by the solution because of the focus on embedding an intelligent capability.

The degree of intelligence also influences the degree of autonomy. It is suggested that there exists a proportional relationship between the two; the greater the degree of intelligence, the greater the degree of autonomy. It is perhaps this reason that intelligent agents are seen as a way to realise the properties of agents as set out by Jennings [JEN01], one of which is autonomy. Autonomy is also a highly desirable property in the context of the problem and the implemented solution. Human involvement is neither necessarily possible nor useful in determining if another agent or host is trustworthy.

As is evident with the criteria of intelligence and autonomy, there may be links between identified criteria. This is the case when one considers criteria such as robustness and reliability on the one hand, and security on the other. Nonetheless, considering only the robustness and reliability of the solution does not translate into analysing the security it provides. Therefore, it is necessary to explicitly consider the security provided by the implemented solution.
This section briefly laid down suitable criteria for analysis of the implemented solution. The section that follows analyses the implemented solution according to the established criteria.

### 13.3 A Qualitative Analysis

The implemented solution is expected to be robust (i.e. resistant to deceptive agents). The robustness expectation stems from the consolidated approach used. Several distinct techniques have been employed including relationship analysis. The consequence of the consolidated approach is that where some techniques may prove ineffective, others are still effective. Thus, in such cases the agent is still able to reach a reasonable estimation of trustworthiness. The extent of the robustness will become clearer in the next section where quantitative results are presented.

The solution in question is also expected to be reliable. That is to say that a client agent should be able to rely on accurate estimates provided by a trust evaluator agent. The reliability of the solution could be seen as stemming from the consolidated approach as a whole. However, it is possible that in some cases the use of several techniques may lead to less accurate estimates. Consider the possibility of a competitor that is in fact trustworthy and behaves ethically. Reputation information and direct experience may indicate the trustworthiness of the agent. The use of relationship analysis however, will indicate otherwise. Nonetheless, in general situations where reputation information is not available and direct experience is minimal, relationship analysis may lead to more accurate estimates. It is thought that the reliability of the solution stems primarily from the principles on which the solution is based. In particular, the decomposition of trust into its constituents along with the attribution of success/failure is thought to be significant. The true reliability of the solution will become more apparent with the presentation of quantitative results in section 13.5.

As far as the adaptability is concerned, the solution is thought to be robust in a changing environment. Again, the use of several techniques ensures that the trust evaluator agent is never in a situation where it is unable to reach an evaluation. If for example, the trust evaluator agent is in an environment where there are a lack of other
trust evaluator agents (reputation providers) the trust evaluator agent still relies on
other techniques to reach an evaluation. The adaptability of the trust evaluator agent is
also dependant on all other agent’s details ‘appearing in the yellow pages’. That is to
say if other agent’s are described by the application domain ontology then the
relationship analysis can always be used. Therefore, the adaptability of the trust
evaluator agent is improved so long as all new agents to the environment register their
descriptions in the application domain ontology. In theory the adaptability of the
agent is further proven by its theoretical ability to provide evaluations for new client
agents, with respect to new target agents, and with respect to new or previously
unknown goals.

Perhaps the one disadvantage to a consolidated approach to trust evaluation is that
more processing must occur to reach an evaluation. The consolidated approach is not
the most efficient in terms of time and space requirements. If analysed at the
component level, the trust evaluation component and the reputation processing
component are all relatively efficient. The attribution component relies on a binary
tree data structure that increases memory requirements. The only real performance
impediment is due to the relationship analysis component. The use of a graph data
structure, the Jena API and the application domain ontology result in increased time
and space requirements. Nonetheless, in a real-world implementation with trust
evaluator agents distributed throughout an MAS (as is intended), the performance is
not expected to be particularly problematic.

In terms of the generality of the solution, it is evident that there has been a tendency to
avoid customisation of the solution to suit a particular application domain. There is
greater utility in producing a generic solution that can be customised as necessary to
suit any given application domain. An example of the customisation of the solution
for a telemanufacturing application can be found in the paper by Pike, Ehlers, and
Oosthuizen [PIK08]. The generality of the solution stems from the use of reusable
components as well as the service orientation of the trust evaluator agent. Arguably,
the only real limitation to the generality of the solution is the use of a domain specific
ontology for the purposes of relationship analysis. Nonetheless, the ontology is

---

20 Although new agents and new goals are catered for, the ability to handle evaluation with respect to
them was not fully implemented in the current version of the prototype.
merely an artefact used by the relationship analysis component and is easily substituted.

The solution is clearly flexible; the trust evaluator agent is able to evaluate the trustworthiness of both software agents and human agents. The trust evaluator agent is also able to evaluate the trustworthiness of the environment. In theory, any service or other entity (as long as it can be modelled as an agent) can be evaluated.

The solution is also believed to be scalable. In fact, it is possible that the addition of more trust evaluator agents to a multi-agent system may improve the performance of the system in terms of reliability. More trust evaluator agents translate to a larger ‘web of trust’ which in turn translates to greater propagation of reputation information. As long as the reputation information being propagated is accurate, a larger web-of-trust seems desirable. The appropriate number of trust evaluator agents for any given environment is difficult to determine. A good deal of experimentation would be required to determine the optimal number of trust evaluator agents.

Given the analysis of the generality, flexibility, and scalability of the solution it can be concluded that the solution is in fact reusable. The papers by Pike et al promote further confidence in the reusability and applicability of the solution [PIK07, PIK08]. Again, it is thought that the reliance on component-based design and the service orientation of the solution promote reusability.

The intelligence criterion poses the question of whether or not the trust evaluator agent behaves rationally. The trust evaluator agent does in fact do the right thing given the state description (knowledge) and its beliefs. Therefore the trust evaluator agent is a rational agent. The trust evaluator agent can be further characterised as cognitive and deliberative; the strengths of the agent’s beliefs play an important role in the agent’s deliberative processes. The deliberative processes are provided by the components. With the exception of the relationship analysis component and the Bayesian attribution component, the components do not individually embed complex intelligence. Admittedly, the trust evaluation component, reputation processing component, and attribution component are simple components embedding a basic capability. The intelligence provided by these three components can be reduced to
simple rules. Therefore, when analysing these components in isolation it may be concluded that the agent is not particularly intelligent. However, it is argued that the intelligence of the trust evaluator agent is emergent. The simple deliberative capabilities provided by the components are combined and orchestrated via the meta-plan for trust evaluation to produce the rational behaviour.

The autonomy of the agent is linked to the intelligence of the agent. The agent is intelligent and therefore certainly exhibits a degree of autonomy. The agent exhibits sufficient autonomy to handle requests for reputation information without any human input or involvement. The agent also is able to select its own plans based on a comparison of state descriptions. Where the agent is unable to choose a plan based on the state description, a plan is randomly selected. While the implemented agent exhibits sufficient autonomy for the purposes of trust evaluation, it is unable to spontaneously capitalise on opportunities in the environment. For example, a proactive trust evaluator agent should take the opportunity to request reputation information from other trust evaluator agents when it is not busy handling requests for reputation information. The trust evaluator agent as implemented does not do this, although its design does cater for such a possibility.

It is argued in this section that the trust evaluator agent exhibits a desirable degree of robustness, reliability and intelligence. Such characteristics indicate a positive influence on security in MASs given the purpose of the trust evaluator agent. It is expected that mobile agents will be able to use the service provided by the trust evaluator agent to avoid malicious hosts. In general, it is expected that any agent will be able to use the service provided by the trust evaluator agent to avoid betrayal by any other agent. Therefore, as an add-on to MASs, the trust evaluator agent is expected to augment existing security related mechanisms to result in more secure systems. Nevertheless, given the nature of trust it is clear that uncertainty is inherent. It is obvious that any trust-related approach has its limitations. The extent of the limitations cannot be established through a qualitative analysis alone.

The following section attempts to establish metrics that allow for experiments to determine the limitations of the solution. In establishing these limitations, the value of the contribution will be quantified.
13.4 Metrics for Quantitative Analysis

The results presented in this section attempt to quantify the extent and limitations of the consolidated trust evaluation scheme that has been implemented. In order to do this metrics need to be established. The obvious question at hand is how successful is the trust evaluator agent at identifying untrustworthy agents? Another formulation of the same question is how often is the trust evaluator agent deceived? Yet another formulation of the question is how often does the trust evaluator agent reach the ‘wrong’ verdict?

The ‘wrong’ verdict is defined as a result that is over some pre-defined threshold, where the threshold indicates an acceptable result or a ‘right’ result. The metric is then defined as the percentage of ‘wrong’ evaluations.

Having identified an appropriate metric it is still necessary to identify the characteristics of the target agents and associated threshold values. In order to provide a complete picture of the extent and limitations of the trust evaluator agent’s capabilities it is necessary to consider a range of different target agents. Table 13.1 identifies the characteristics of six different target agents as well as appropriate threshold values for each. Table 13.1 also indicates the techniques (except for the evaluation technique itself) that are likely to be effective or ineffective in the evaluation.

**Agent 1** is a malicious agent the malicious actions of which are undetectable by the client agent. An example of such an agent is a host controller agent in a mobile agent system that reads or makes unauthorised copies of mobile agents at the remote host. Such an action is undetectable to the mobile agent or another agent at another host. The reputation information as provided by a single simulated trust evaluator agent is inflated for agent 1. The trust evaluator agent carrying out the evaluation places little credibility in the reputation information. There is no relationship between the client agent and the target agent. The malicious actions of agent 1 do not necessarily result in failure of the delegate goal. For example, the host controller agent may facilitate the execution of the mobile agent, but still make a copy of the mobile agent thus constituting betrayal. Attribution of the success/ failure of the delegated goal is
<table>
<thead>
<tr>
<th>Agent</th>
<th>Characteristics</th>
<th>Reputation</th>
<th>Relationship</th>
<th>Threshold</th>
<th>Effective Techniques</th>
<th>Ineffective Techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Malicious: Undetectable malicious action. May or may not result in overall failure of the delegated goal.</td>
<td>Elevated reputation information, low credibility</td>
<td>None</td>
<td>Medium Low Trust</td>
<td>Attribution (partial), Reputation, Relationship,</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Malicious: Undetectable malicious action. May or may not result in overall failure of the delegated goal.</td>
<td>Elevated reputation information, low credibility</td>
<td>Competitor</td>
<td>Medium Low Trust</td>
<td>Attribution (partial)</td>
<td>Attribution (partial), Reputation</td>
</tr>
<tr>
<td>3</td>
<td>Malicious: Detectable malicious action. May or may not result in overall failure of the delegated goal.</td>
<td>Elevated reputation information, low credibility</td>
<td>None</td>
<td>Medium Low Trust</td>
<td>Attribution, Reputation</td>
<td>Relationship</td>
</tr>
</tbody>
</table>

Table 13.1 Target Agents Characteristics, Thresholds, and Effective Techniques
<table>
<thead>
<tr>
<th>Agent</th>
<th>Characteristics</th>
<th>Reputation</th>
<th>Relationship</th>
<th>Threshold</th>
<th>Effective Techniques</th>
<th>Ineffective Techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Competence\Disposition: incompetent and/or lacks disposition: constitutes agent betrayal 50% of the time.</td>
<td>Elevated reputation information, low credibility</td>
<td>None</td>
<td>Medium Low Trust</td>
<td>Attribution, Reputation</td>
<td>Relationship</td>
</tr>
<tr>
<td>5</td>
<td>Competence\Disposition: incompetent and/or lacks disposition: constitutes agent betrayal 33% of the time.</td>
<td>Relatively accurate reputation information, low credibility</td>
<td>None</td>
<td>Medium Low Trust</td>
<td>Attribution, Reputation</td>
<td>Relationship</td>
</tr>
<tr>
<td>6</td>
<td>Trustworthy: completely trustworthy, is able to overcome anomalies in the environment.</td>
<td>Slightly deflated reputation information, low credibility</td>
<td>None</td>
<td>Medium Trust</td>
<td>Attribution, Reputation</td>
<td>Relationship</td>
</tr>
</tbody>
</table>

Table 13.1 Target Agents Characteristics, Thresholds, and Effective Techniques
considered to be partially successful. Obviously this particular agent is untrustworthy and it is desirable for the trust evaluator agent’s evaluation to reflect this. Values of low trust and medium low trust (both of which are ontological concepts defined by the trust evaluator agent’s ontology of trust) are considered acceptable.

Agent 2 is a malicious agent and identical to agent 1 as characterised above, with one exception: agent 2 is considered to be in direct competition with the client agent. The trust evaluator agent will therefore be able to rely on relationship analysis as an effective technique for establishing the existence of the relationship.

Agent 3’s characterisation is identical to that of agent 1 and agent 2 with one major difference: the malicious action executed by agent 3 is detectable. For example, agent 3 could be a host controller agent that by tampering with the machine code or data state (inserting virulent code, modifying data state) violates the integrity of the mobile agent. The violation will result in obfuscation or corruption of the agent that is certainly detectable. Since the malicious action is always detectable in this case, attribution is more effective. The use of reputation information may also be more effective in this case because the malicious action is presumed to be detectable by all agents. The usefulness of reputation information is dependant on whether the target agent targets only one client agent with malicious action or all client agents with malicious action.

Agent 4 differs from agents 1 – 3 in that it lacks the competence and/or the disposition. The lack of competence/disposition will translate into agent betrayal with a prior probability of 0.5. That is to say, that the client agent will perceive agent betrayal once in every two delegations of the particular goal. The agent betrayal does not necessarily translate into failure to achieve the goal. It is possible that events in the environment will conspire to ensure achievement of the goal irrespective of the agent betrayal.

Agent 5’s characterisation is identical to agent 4’s characterisation with one notable exception: agent 5 is slightly more competent and/or willing/persistent. Agent 5’s lack of competence or disposition will constitute agent betrayal with a prior probability of 0.33. The same reputation information as used for agents 1 - 4 is reused for agent 5.
Since agent 5 is slightly more competent and/or willing/persistent the reputation information is naturally more accurate. Nonetheless, low credibility is placed in its provider.

**Agent 6** is at the opposite end of the spectrum when compared with agents 1 – 5. Agent 6 is deemed to be highly competent with a favourable disposition. Agent 6 is able to overcome obstacles or anomalies in the environment to achieve the goal. Thus agent 6 never betrays the client agent and always ensures the success of the delegate goal. Naturally, agent 6 is highly trustworthy and this should be reflected in the trust evaluator agent’s evaluation. Thus the threshold should be the concept of medium trust, meaning that evaluation’s resulting in low trust or medium low trust are considered ‘wrong’. The same reputation information as used for agents 1 - 5 is reused for agent 6. Given the trustworthiness of agent 6, the reputation information is slightly deflated, but little credibility is placed in it. The use of the attribution capability and reputation capability are regarded as effective in this case, while reputation analysis is not.

The agents characterised in section 13.4 (and summarised in table 13.1) are representative of agents that may be encountered in many environments. While the list is not exhaustive, it is thought to be the basis for a set of experiments that clearly identify the extent and limitations of the solution. The results of the experiments are presented in the next section.

**13.5 Quantitative Analysis**

Section 13.5 presents the results of six experiments, each corresponding to the evaluation of one of the six agents characterised in the previous section. For each experiment five tests were conducted. The tests are independent of one another and the trust evaluator agent’s beliefs are not revised from one test to another. This means that in each test the trust evaluator agent evaluates the target agent for the client agent from ‘scratch’. Each test involves ten evaluations of the same target agent with respect to the same goal.
The charts that follow plot the result of each evaluation for each test for each experiment. For each experiment a separate chart is also provided, plotting the average results (across the five tests). The result of an evaluation is a number between 1 and 5. The numbers 1 to 5 correspond to the concepts low trust (1), medium low trust (2), medium trust (3), medium high trust (4), and high trust (5).

Chart 13.1 displays the results of the experiment for target agent 1. The values clearly range between low trust and high trust. There is little consistency in the results between tests. The inconsistency can be attributed to two variables; whether or not the goal was achieved successfully and whether or not environmental anomalies were perceived at the time of delegation.

When interpreting the results, it should also be remembered that the trust evaluator agent provides an evaluation indicating its trust that the goal will be achieved. The trust in the target agent only forms part of the trust that the goal will be achieved.

The average results for experiment 1 are displayed in chart 13.2. It is clear from these results that the trust evaluator agent (despite a promising start) provides many ‘wrong’
evaluations; too many evaluations indicating medium high trust and high trust are given to the client agent. Such evaluations are to be expected and are of no real fault of the solution itself; the problem stems from the inability to perceive certain malicious actions.

Chart 13.2 Average Results for Experiment 1

The results of experiment 2 (involving target agent 2) can be seen in chart 13.3. The most notable difference in the results compared to experiment 1 is that the concept of high trust is never reported. In fact, the concept of medium high trust is only reported once, and the concepts of medium trust are only reported a handful of times. The results are generally lower and thus an improvement from the results of experiment 1. This is attributed to the effectiveness of relationship analysis in establishing the existence of the competitor relationship.

Chart 13.4 illustrates the average results of experiment 2. These results again reflect the value of relationship analysis. The average result is seldom above medium low trust (2), compared with the average results from experiment 1 where the results are seldom below medium trust (3).
Experiment 3 considers a malicious agent whose malicious actions are detectable. Chart 13.5 illustrates the results of experiment 3. It is clear from the chart that the
trust evaluator agent reaches evaluations of medium trust more often than is desirable, given the malicious nature of agent 3. However, it is worth putting the results in context. Despite initial fallibility in test 1 the trust evaluator agent’s last seven evaluations are acceptable. In test 3 the trust evaluator agent appears to consistently reach the ‘right’ results in the later half of the test. In test 4 the trust evaluator agent performs remarkably well, without ever reaching a ‘wrong’ result. It should also be pointed out that the results never reach the heights of medium high trust or high trust.

So, to what can the undesirable performances seen in tests 1 and 5 be attributed to? It must be remembered that the trust evaluator agent provides an estimate of the trust that the goal will be achieved. It is plausible (and in fact likely) that the results of medium trust stem from the prior achievement of the goal due to external factors in the environment.

The average results as illustrated in chart 13.6 are also of interest. On average the results are always below medium trust (3), in stark contrast to the average results of experiment 1. Also it appears that there is greater consistency in the last three evaluations with results tending towards medium low trust and low trust.

### Chart 13.5 Experiment 3 – Agent 3

![Chart 13.5 Experiment 3 – Agent 3](image-url)
Experiment 4 tests the trust evaluator agent given an incompetent or unwilling target agent in agent 4. Chart 13.7 shows the effective performance of the trust evaluator agent across the five tests. The results of evaluations peak at medium high trust and medium trust. Such high evaluations are most likely due to the occasional success of the target agent possibly in cases where the target agent overcomes obstacles in the environment.

In reality it is unlikely that the target agent is able to overcome obstacles in the environment given its incompetence and/or lack of willingness and perseverance. Therefore it is thought that such high results only occur due to the rather trivial simulation of the target agent (based on random numbers for the various outcomes).

Chart 13.8 illustrates the average results for experiment 4. The average results show the effectiveness of the trust evaluator agent. On average, it is only the result of the initial evaluation that is above medium low trust. Initial evaluations are always expected to be problematic given the possible lack of evidence. Lower initial values for the trust evaluator agent’s starting beliefs, corresponding to a cautious and risk averse agent would alleviate the problem in this case.
Chart 13.7 Experiment 4 – Agent 4

Chart 13.8 Average Results for Experiment 4
Experiment 5 tests the trust evaluator agent with target agent 5. Target agent 5 is very similar to target agent 4 with one notable exception. Target agent 5’s efforts only constitute betrayal with a prior probability of 0.33. Intuitively target agent 5 is expected to betray the client agent less often than target agent 4. The consequence of this is that there will probably be fewer situations where the target agent betrays the client agent and the goal is not achieved. In cases where agent betrayal and failure to achieve the goal coincide, the trust that the goal will be achieved in the future is dealt a serious blow. Since such cases are thought to be less likely, slightly higher results (compared to experiment 4) are thought to be reasonable.

The results of experiment 5 are shown in chart 13.9. The results fluctuate between low trust and medium high trust. The fluctuations can be attributed (at least indirectly) to the unpredictability of the target agent. As can be seen from the chart the trust evaluator agent never reaches an evaluation of high trust and only once reaches an evaluation of medium high trust. The result of medium high trust occurs relatively early on in proceedings (the second evaluation of test 2). Thereafter the results are not particularly high indicating the unpredictability that the goal will be achieved.

<table>
<thead>
<tr>
<th>Number of Evaluations</th>
<th>Test 1</th>
<th>Test 2</th>
<th>Test 3</th>
<th>Test 4</th>
<th>Test 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

Chart 13.9 Experiment 5 – Agent 5
The average results of experiment 5 as reflected in chart 13.10 show relatively low results. Nonetheless, the average results of the evaluations for experiment 5 are higher than the average results of the evaluations for experiment 4. The average results for experiments 4 and 5 are plotted on the same chart (chart 13.11) to make comparison easier. In seven of the ten evaluations higher values of trust (or at least the same values of trust) were reported for experiment 5 involving target agent 5. Given agent 5’s lower prior probability of betrayal the results seem plausible.

![Chart 13.10 Average Results for Experiment 5](chart.png)

Experiment 6 tests the trust evaluator agent with an unequivocally trustworthy target agent. Such an experiment is necessary to establish that the trust evaluator agent provides fair evaluations and does not have some sort of bias. The results of the experiment can be seen in chart 13.12.

The results are appropriately high and remarkably consistent. The trust evaluator agent reports medium trust for the first evaluation of each test. Thereafter there is an increase in the results as the agent’s beliefs are revised.
Chart 13.13 shows the average results of the experiment. These results prove the trust evaluator agent’s efficacy in evaluating trustworthy agents.
The results presented so far in this section are summarised in chart 13.14 that also serves to summarise the performance of the trust evaluator agent. The chart shows the level of deception for each experiment. That is, the chart shows deception as a percentage of the total number of evaluations (50 evaluations over five independent tests) for each experiment.

Deception is defined as a result that is over the threshold value in the unintended direction. For example, in the case of an untrustworthy agent with a trust threshold of medium low trust all results of medium trust and higher constitute a deception of the trust evaluator agent. Similarly, in the case of a trustworthy agent with a trust threshold of medium trust all results less than that constitute deception.

Chart 13.14 shows that the trust evaluator agent is deceived 50% of the time in experiment 1. The deception is attributed to the inability to perceive the malicious action of target agent 1 rather than any incompetence in the trust evaluator agent’s capability. As far as experiment 1 is concerned, the trust evaluator agent performs as least as well as a guessing trust evaluator agent that randomly decides on the trustworthiness of the target agent with a prior probability of 0.5.
In experiment 2 the trust evaluator agent performs much better; it is only deceived 18% of the time. A comparison with the level of deception experienced in experiment 1 indicates the value of relationship analysis in particular and a consolidated approach in general.

The trust evaluator agent is deceived 24% of the time in experiment 3. The level of deception is marginally higher than that of experiment 2. The difference is somewhat surprising given the malicious nature of agent 3’s actions. The difference is attributed to the effectiveness of relationship analysis in experiment 2. The result is also indicative of the need to fine-tune the belief revision within the attribution component. Such fine-tuning would be an experimental process. A more advanced belief revision process may also prove beneficial.

The level of deception as far as agent 4 is concerned is 14%. The result shows the effectiveness but also the limitations of the solution. The trust evaluator agent will
certainly minimise incidents of betrayal (its goal), but will seldom be able to eliminate them completely.

The results of experiment 5 are indicative of increased levels of deception compared with the results of experiment 4. However, it must be remembered that target agent 5 will betray the client agent less often. Furthermore, the trust threshold for both experiments remains the same. Therefore, the increased level of deception in experiment 5 (26%) is attributed to the increased trustworthiness of the target agent and the fact that the trust threshold remains low. The result is therefore merely further evidence of the rationality of the trust evaluator agent.

Further evidence of the rationality of the trust evaluator agent is provided by the result of experiment 6. As the chart shows, the trust evaluator agent is never deceived. The trust evaluator agent is able to correctly report the trustworthiness of a trustworthy target agent.

If experiment 1 is ignored, the results show that the trust evaluator agent is deceived less than 33% of the time. The results show both the efficacy and limitations of the solution. The effectiveness and limitations of the solution are summarised in the concluding section.

13.6 Conclusion
The results provided in the previous section are promising especially when placed in context. The trust evaluator agent does not overly rely on reputation information in the experiments. However, if a web of trust existed amongst trust evaluator agents through which credible reputation information could be gleaned then improved results could be expected. Furthermore, the tests that comprise the experiments span ten evaluations for practicality sakes. The number of evaluations is acceptable given that REVALUATOR is primarily intended as proof of concept for the trust evaluator agent. It is expected that the accuracy and consistency of the results will improve over many more evaluations (a question for future research). It is reasonable to expect that the trust evaluator agent’s evaluations will converge (become consistent) as a state of
equilibrium is reached in the environment, assuming consistent target agent behaviour and sufficient experience and information for the trust evaluator agent.

On the other hand, it is clear that the trust evaluator agent does have its limitations. The efficacy of the trust evaluator agent is also clearly dependent on the perceptive abilities of the client agents. If the client agent is able to perceive a lack of disposition or a malicious action for example, then the trust evaluator agent’s individual beliefs can be revised more accurately to produce more realistic evaluations.

The quantitative analysis of the results provided in the previous section supplements the qualitative analysis of section 13.3. It is clear that the trust evaluator agent does offer some resistance to deceptive agents. Furthermore, the results also show that the trust evaluator agent provides accurate and reliable reputation reports. The results also attest to the rationality of the trust evaluator agent. Therefore the results further prove that an intelligent trust evaluation capability (as provided by the respective components) has been successfully embedded into the agent.

Chapter 13 has provided results and analysis of the implemented prototype and the solution in general. The analysis of the solution provided in this chapter can be seen as a starting point for future work. Potential avenues for future research are considered in the final chapter that also concludes the dissertation.
Chapter 14
Conclusion and Future Work

14.1 Introduction
The final chapter of the dissertation reflects on the contribution of the work and briefly highlights potential avenues of future research. Chapter 14 will outline the contribution of each chapter comprising the dissertation. However, it is also important to take a holistic perspective when analysing the contribution of the dissertation. Focusing on the contribution of the dissertation is not only important for the means of summary. It is a necessary precursor to establishing areas for future research.

The chapter is arranged as follows: section 14.2 addresses the contribution of each chapter, summarising the main ideas presented in each. Section 14.3 takes a holistic look at the contribution of the work. Section 14.4 considers future work that could be undertaken. Section 14.5 concludes the chapter with a brief summary.

14.2 Contribution
The introductory chapter of the dissertation outlines several problems. Several questions are posed: How can agents trust one another? How can agents trust agent hosts? The main purpose of the research was to investigate and develop an intelligent means aimed at protecting mobile agents in particular, and agents in general. This section indicates in which chapters these questions have been answered and where the purpose of the research has been fulfilled by reflecting on the contribution of each chapter.

Chapter 2 introduced fundamental concepts of agency. The chapter presents the concept of an agent. The chapter also pays attention to the idea of an intelligent or rational agent. The concept of mobility is addressed and hence the concept of a mobile agent is presented. Mobile agent systems are also introduced. The chapter provides the foundation for the rest of the dissertation.
Chapter 3 also aims at establishing a foundation. A theoretical foundation is laid by
analysis of the literature relating to the concept of trust. The chapter defines trust and
discusses the semantics of trust. The chapter describes trust as a belief. Trust therefore
implies inherent uncertainty. Therefore, trust involves an acceptance of risk and
relinquishing of control. Perhaps the most important contribution of the chapter is the
decomposition of trust into a set of connected beliefs. Sources of belief and therefore
trust are also established. The concept of delegation is also introduced. Different types
of trust are highlighted. Attribution of the outcome of delegation is also discussed.
The chapter also pays attention to more theoretical concepts such as how trust
influences trust. The concept of control also receives attention. The chapter clearly
reveals how agents may determine whether other agents are trustworthy. There is
evidence of the application of these concepts, as presented in chapter 3, in the
REVALUATOR prototype produced as part of the research.

Chapter 4 looks at various approaches to evaluating trustworthiness of agents. The
consideration of the existing approaches forms part of a broader investigation into the
state of the art. A cognitive model for trust evaluation based on the theory of chapter 3
is discussed, along with a statistical model, another probabilistic model, a reputation
focused approach, and a model for trust in open systems. Arguably, the most salient
contribution of the chapter is the criteria for analysing the surveyed approaches to
trust evaluation. The same criteria are used in analysing the implemented solution in
chapter 13.

Chapter 5 continues the investigation into the state of the art. The importance of an
ontology for reputation is highlighted. Social network analysis and in particular
relationship analysis are presented as a methodology to improve trust evaluation. An
incentive compatible reputation mechanism is presented as a means to improve the
elicitation of reputation information and the information itself.

Chapters 3 to 5 show how an agent can evaluate the trustworthiness of another agent
and then reach a rational decision to trust it. Chapter 6 diverges from the topic of trust
and reputation to instead focus on appropriate architectural models for agents. Four
existing architectural models for general BDI-theoretic agents are analysed. Architectural abstractions suitable for designing agent capable of trust are identified.
The focus of the work again shifts slightly with chapter 7 to consider several mobile agent systems. Criteria for comparing the surveyed mobile agent systems are introduced. The mobile agent systems are compared using the criteria as a basis for comparison. The chapter concludes that none of the surveyed mobile agent systems offer adequate protection for mobile agents, further motivating the value of the dissertation research.

The remainder of the dissertation focuses on developing a trust evaluation capability to be embedded into agents to enhance agent protection. Chapter 8’s contribution is an architectural model for a trust enhanced mobile agent system that shows the addition of trust evaluator agents as a means to enhance agent protection through the provision of a trust evaluation service. The architectural model for the trust evaluator agent is outlined in chapter 9.

Chapter 10 builds on the architectural model of chapter 9 by detailing the plans to be executed by the trust evaluator agent in continual pursuit of its goal to minimise incidents of betrayal in the environment. Chapter 10 constitutes a high level outline of a consolidated trust model that employs different techniques to reach evaluations. The model consolidates the use of relationship analysis, reputation information, attribution and a cognitive approach to trust evaluation.

Chapter 11 builds on chapter 10 by presenting and analysing the high level design of five components that embed the aforementioned capabilities into the trust evaluator agent. The deliberative processes provided by the components are bootstrapped by the plans presented in chapter 10.

Chapters 8 to 11 constitute a solution to the problem outlined in the introduction. Chapter 12 describes the lower level design and implementation of the solution. The chapter provides insight into the design and development of the REVALUATOR prototype. The chapter is the result of an extensive effort proving the viability of the approach.

The penultimate chapter of the work provides both a qualitative and quantitative analysis of the implemented solution. The chapter presents results that clearly show
both the strengths and the limitations of the solution. The contribution of the solution is elaborated upon in the following section.

14.3 The Holistic Contribution of the Solution

The benefits of the solution lie in the possibility to add trust evaluator agents to existing mobile agent systems (and multi-agent systems in general) as a means to augment existing mechanisms for agent protection. The continued use of useful systems is ensured by the addition of trust evaluator agents. The addition of trust evaluator agents limits the re-engineering of useful systems in principle at least.

The trust evaluator agents also introduce a degree of desirable service orientation that promotes heterogeneity and provides an abstraction to be exploited. The service provided by the trust evaluator agent unquestionably enhances agent protection. However, as shown in chapter 13, there are limits to this protection. The solution does not ensure or guarantee the total safety of client agents. Due to the very nature of trust, specifically the inherent uncertainty involved, it would seem that complete prevention and avoidance of betrayal is impossible - at least in most environments. Therefore, the solution does not enable the application of mobile agent systems in security critical domains (but then again, neither does the use of existing security mechanisms). Clearly, traditional client-server models are more suited to such domains.

Nonetheless, there are numerous application domains that could benefit from this work. Any domain involving social relations between agents potentially stands to benefit. Applications involving agent negotiation (such as agent marketplaces) are one such example.

The developed evaluation capability is also useful in terms of the broader development of social agents. Trust is a social phenomenon. If it is desirable to develop agents with social abilities then the developed capabilities (provided as reusable components) can be embedded into agents to aid in the provision of such social capabilities.
Section 14.3 has analysed the contribution of the solution from a holistic perspective. The following section outlines areas for future work.

### 14.4 Future Work

The REVALUATOR prototype could possibly be extended. The relationship analysis component could be extended to handle a variety of relationships, not just the competitor relationship. The relationship analysis component could also benefit from the implementation of a previously outlined optimisation (relationship graph). The reputation processing component could possibly be improved incorporating more advanced features. Further development of the trust evaluator agents’ ‘infrastructure’ would also be useful in facilitating future work. In particular, the agent’s knowledge structures (state description), communicative abilities and controller could benefit from further improvement.

In terms of the multi-agent systems aspects of the solution, various extensions are obvious. The further development and deployment of client, target and trust evaluator agents would be useful in further assessment of the presented solution.

At a more holistic level, it is unclear precisely how much utility exists in further work on the topic of trust and reputation. Trust is inherently uncertain and there is no escaping occasional betrayal in certain circumstances. Therefore, the most logical topic for future study in as far as trust is concerned is the idea of contractual trust – or third parties trust. The development of a prototype involving not only client, target, and trust evaluator agents, but also arbiter agents that enforce contracts seems an avenue worthy of exploration.

In the same vein, the implementation of some of the more advanced theoretical topics summarised in chapter 3 remains an option. However, such work would have to be carried out taking into consideration the cognitive and psychological abilities of the participating agents. Some of the concepts presented in chapter 3 would only be useful given highly advanced intelligent agents with advanced cognitive abilities.
In terms of supplementary contributions to trust models, the development of useful ontologies is also required. The further development of an ontology for reputation information is a potential area for future work. A more advanced ontology to support the trust evaluation service as provided by trust evaluator agents would also be beneficial.

At the analytical level, establishing universally accepted qualitative criteria and quantitative metrics for the purposes of comparing trust models seems sensible.

This section highlights some potential areas for future work. The section that follows concludes the chapter.

14.5 Conclusion
Starting with a brief summary of the contribution of each chapter in this work, the chapter examines the contribution of the work holistically. The chapter also briefly examines areas worthy of future research.

It is clear that this work answers the questions posed in the first chapter of this work. More importantly, a solution to the problem outlined in that chapter has been presented and implemented. Therein lies the contribution of the work; an intelligent and general solution that applies the principles of agent trust to protecting agents from each other, and importantly, from agent hosts.

Additionally, the work contributes to the broader topic of social agency. The dissertation research can also be seen in the broader context as ‘a step amongst the stairs’ leading to even more advanced and interesting agent-oriented applications.
Appendix A

REVALUATOR Static Structure Diagrams

A.1 JADE – Specialised Trust Evaluator Agent Structural Relationship 253
A.2 Specialised Trust Evaluator Agent Static Structure 254
A.3 Specialised Trust Evaluator Agent Ontology Static Structure 255
A.4 Data Structures Static Structure 256
A.5 Trust Evaluation Component Static Structure 257
A.6 Attribution Component Static Structure 258
A.7 Bayesian Attribution Component Static Structure 259
A.8 Relationship Analysis Component Static Structure 260
A.9 Reputation Component Static Structure 261
A.10 Client Agent Static Structure 262
A.1 JADE – Specialised Trust Evaluator Agent Structural Relationship
A.2 Specialised Trust Evaluator Agent Static Structure
A.3 Specialised Trust Evaluator Agent Ontology Static Structure
A.4 Data Structures Static Structure
A.5 Trust Evaluation Component Static Structure
A.6 Attribution Component Static Structure
A.7 Bayesian Attribution Component Static Structure
A.8 Relationship Analysis Component Static Structure
A.9 Reputation Component Static Structure
A.10 Client Agent Static Structure
Appendix B

Ontologies

B.1 Trust Service Communication Ontology 264
B.2 Telemanufacturing Service Ontology (including instances) 265
B.3 Telemanufacturing Service Ontology (Classes and Properties only) 266
B.1 Trust Service Communication Ontology
B.2 Telemanufacturing Service Ontology (including instances)
B.3 Telemanufacturing Service Ontology (Classes and Properties only)
References


