

**MODELLING AND REASONING ABOUT TRUST  
RELATIONSHIPS IN THE DEVELOPMENT OF  
TRUSTWORTHY INFORMATION SYSTEMS**

**Michail Pavlidis**

**Ph.D.**

**2014**

**MODELLING AND REASONING ABOUT TRUST  
RELATIONSHIPS IN THE DEVELOPMENT OF  
TRUSTWORTHY INFORMATION SYSTEMS**

**Michail Pavlidis**

**A thesis submitted in partial fulfilment of the requirements of the School of  
Architecture, Computing, and Engineering, University of East London for  
the degree of Doctor of Philosophy**

**October 2014**

# Abstract

*Trustworthy information systems are information systems that fulfil all the functional and non-functional requirements. To this end, all the components of an information system, either human or technical, need to collaborate in order to meet its requirements and achieve its goals. This entails that system components will show the desired or expected behaviour once the system is put in operation. However, modern information systems include a great number of components that can behave in a very unpredictable way. This unpredictability of the behaviour of the system components is a major challenge to the development of trustworthy information systems and more particularly during the modelling stage. When a system component is modelled as part of a requirements engineering model it creates an uncertainty about its future behaviour, thus undermining the accuracy of the system model and eventually the system trustworthiness. Therefore, the addition of system components inevitably is based on assumptions of their future behaviour. Such assumptions are underlying the development of a system and usually are assumptions of trust by the system developer about her trust relationships with the system components, which are instantly formed when a component is inserted into a requirements engineering model of a system. However, despite the importance of such issues, a requirements engineering methodology that explicitly captures such trust relationships along with the entailing trust assumptions and trustworthiness requirements is still missing.*

*For tackling the preceding problems, the thesis proposes a requirements engineering methodology, namely JTrust (Justifying Trust) for developing trustworthy information systems. The methodology is founded upon the notions of trust and control as the means of confidence achievement. In order to develop an information system the developer needs to consider her trust relationships with the system components that are formed with their addition in a system model, reason about them, and proceed to a justified decision about the design of the system. If the system component cannot be trusted to behave in a desired or expected way then the question of what are the alternatives in order to build confidence in the future behaviour of the system component raises. To answer this question we define a new class of requirements, namely trustworthiness requirements. Trustworthiness requirements prescribe the functionality of the software included in the information system that compels the rest of the information system components to behave in a desired or expected way. The proposed methodology consists of: (i) a modelling language which contains trust*

*and control abstractions; (ii) and a methodological process for capturing and reasoning about trust relationships, modelling and analysing trustworthiness requirements, and assessing the system trustworthiness at a requirements stage. The methodology is accompanied by a CASE tool to support it.*

*To evaluate our proposal, we have applied our methodology to a case study, and we carried out a survey to get feedback from experts. The topic of the case study was the e-health care system of the National Health Service in England, which was used to reason about trust relationships with system components and identify trustworthiness requirements. Researchers from three academic institutions across Europe and from one industrial company, British Telecom, have participated in the survey in order to provide valuable feedback about the effectiveness and efficiency of the methodology. The results conclude that JTrust is useful and easy to use in modelling and reasoning about trust relationships, modelling and analysing trustworthiness requirements and assessing the system trustworthiness at a requirements level.*

# Dedication

To my parents Ioannis and Anastasia Pavlidis  
For your unconditional love and support throughout my life

I thank you

Στους γονείς μου Ιωάννη και Αναστασία Παυλίδη

Σας ευχαριστώ



## Acknowledgements

I am extremely grateful to my supervisor Haris Mouratidis, whose vision and enthusiasm have guided and encouraged my research over the past years. I thank him for opening up for me a new world, new perspectives, and new expectations. It was a privilege to have such outstanding supervisor.

I would like to give special thanks to my second supervisor Shareef Islam for his timeless and continuous guidance, advice, and encouragement while carrying out this research. I owe special thanks to my external advisor Paul Kearney for his exceptional support and for sharing his wealth of knowledge. Our discussions gave me many insights, invaluable advice, and suggestions for my research.

I would like also to thank all the researchers from University of East London, British Telecom, University of the Aegean, and University of Castile-La Mancha that took part in the evaluation survey and for their feedback and very useful discussions for improving my thesis. I would like also to give special thanks to Piotr Cofta.

I would also like to express my thankfulness to the staff at School of Architecture, Computing and Engineering (ACE) and Graduate School at University of East London (UEL) for their support in my research. My thankfulness goes also to staff at British Telecom for providing me time for discussion on my research.

Gratitude is shown to British Telecom (BT) and Engineering and Physical Sciences Research Council (EPSRC) for their funding with regards to this research.

Last but not least, I would like to thank my beloved parents Ioannis and Anastasia, and my sister Magdalini, for being patient, encouraging, understanding, and extremely supportive and for their unconditional love not only throughout my PhD but also throughout my life.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation and problem statement . . . . .	3
1.2	Research questions . . . . .	6
1.3	Research aims and objectives . . . . .	6
1.4	Research contributions . . . . .	7
1.5	Research approach . . . . .	9
1.6	Publications . . . . .	11
1.7	Structure of the thesis . . . . .	13
<b>I</b>	<b>State of the Art</b>	<b>16</b>
<b>2</b>	<b>Literature review</b>	<b>17</b>
2.1	Trust and trustworthiness . . . . .	18
2.2	Information System trust and trustworthiness . . . . .	24
2.3	Trust and Information System trustworthiness in the context of the thesis . . . . .	28
2.4	Information Systems development methodologies . . . . .	29
2.4.1	Modelling language . . . . .	30
2.4.2	CASE tools for Information System methodology . . . . .	32
2.5	Evaluation methods for software engineering methodologies . . . . .	34
2.6	State of the art in trust engineering . . . . .	35
2.6.1	Trust modelling . . . . .	36
2.6.2	Security engineering considering trust . . . . .	43
2.6.3	Goal satisfaction reasoning . . . . .	50
2.6.4	Trust management . . . . .	53
2.6.5	Human Computer Interaction in the context of trust . . . . .	57
2.6.6	Trusted Computing . . . . .	58
2.6.7	Computational trust . . . . .	59



2.7	Chapter summary . . . . .	59
<b>II JTrust: A Trustworthy Information System Development Methodology</b>		<b>63</b>
<b>3</b>	<b>JTrust modelling language</b>	<b>64</b>
3.1	Methodology requirements . . . . .	65
3.2	Methodology structure . . . . .	66
3.3	Running example . . . . .	66
3.4	Confidence as the key to modelling uncertainty . . . . .	68
3.5	JTrust modelling language concepts . . . . .	75
3.6	Trustworthiness requirements . . . . .	82
3.7	Meta-model of the JTrust modelling language . . . . .	83
3.8	Trustworthiness assessment model . . . . .	90
3.9	Chapter summary . . . . .	92
<b>4</b>	<b>JTrust process</b>	<b>93</b>
4.1	Activity 1: Goal and dependency modelling . . . . .	94
4.2	Activity 2: Resolution modelling . . . . .	96
4.3	Activity 3: Entailment modelling . . . . .	98
4.4	Activity 4: Trustworthiness requirement analysis . . . . .	100
4.5	Activity 5: System trustworthiness assessment . . . . .	101
4.6	Chapter summary . . . . .	102
<b>5</b>	<b>JTrust tool</b>	<b>104</b>
5.1	Tool architecture . . . . .	104
5.2	Concepts graphical notation . . . . .	107
5.3	Trust tool functionality . . . . .	111
5.4	Chapter summary . . . . .	113
<b>III Evaluation and Conclusions</b>		<b>114</b>
<b>6</b>	<b>Evaluation</b>	<b>115</b>
6.1	Study 1: Evaluation of JTrust by case study research in the health care domain . . . . .	117
6.1.1	Case study design . . . . .	119
6.1.2	Data collection - Applying the JTrust methodology . . . . .	125
6.1.3	Data analysis - Evaluation results and discussion . . . . .	140

6.2	Study 2: Evaluation of JTrust methodology by survey research . . . .	144
6.2.1	Survey design . . . . .	145
6.2.2	Data collection . . . . .	151
6.2.3	Data analysis - Evaluation results and discussion . . . . .	152
6.3	Chapter summary . . . . .	156
<b>7</b>	<b>Conclusions and future work</b>	<b>157</b>
7.1	Thesis summary . . . . .	158
7.2	Research questions . . . . .	159
7.3	Contributions to the state of the art and impact . . . . .	161
7.4	Limitations of the approach . . . . .	163
7.5	Future work . . . . .	164
	<b>Bibliography</b>	<b>167</b>

# List of Tables

- 2.1 Trust engineering approaches . . . . . 36
- 2.2 Comparison table of state of the art in trust engineering . . . . . 61
  
- 7.1 Summary of research questions and answers . . . . . 160

# List of Figures

1.1	Research approach . . . . .	10
1.2	Thesis claim and supporting propositions . . . . .	15
2.1	Optimum trust level . . . . .	23
2.2	Modelling methods components . . . . .	31
2.3	Monitoring example . . . . .	38
2.4	Bimrah meta-modell . . . . .	40
2.5	Methods for bridging trusted domains . . . . .	41
2.6	Holistic trust analysis process . . . . .	42
2.7	Trust case conceptual model . . . . .	45
2.8	CORAS' steps . . . . .	49
2.9	Trust reference model . . . . .	56
2.10	Application areas of state of the art in trust engineering . . . . .	62
3.1	JTrust methodology structure . . . . .	66
3.2	Model of confidence . . . . .	72
3.3	JTrust modelling language meta-model . . . . .	84
3.4	Goal model example . . . . .	85
3.5	Dependency model example . . . . .	87
3.6	Resolution model example . . . . .	88
3.7	Entailment model example . . . . .	89
3.8	Trustworthiness requirement model example . . . . .	90
4.1	JTrust process . . . . .	94
4.2	Goal and dependency modelling activity . . . . .	96
4.3	Resolution modelling activity . . . . .	98
4.4	Entailment modelling activity . . . . .	99
4.5	Trustworthiness requirement analysis activityl . . . . .	100
4.6	System trustworthiness assessment activity . . . . .	102
5.1	Implemenation meta-model . . . . .	105

5.2	Model driven architecture . . . . .	106
5.3	JTrust tool visual layout . . . . .	106
5.4	Concepts notation . . . . .	107
5.5	Actor properties . . . . .	108
5.6	Goal Properties . . . . .	109
5.7	Decomposition Properties . . . . .	109
5.8	Trust resolution Properties . . . . .	110
5.9	Control Resolution Properties . . . . .	110
5.10	Entailment Properties . . . . .	111
5.11	Dependency Properties . . . . .	111
6.1	Evaluation approach . . . . .	117
6.2	e-Health scenario . . . . .	122
6.3	Case study collection methods . . . . .	123
6.4	Summary care record of a patient . . . . .	129
6.5	NHS System goal diagram . . . . .	130
6.6	Modifying the capability property of a goal . . . . .	131
6.7	Correspondences links between NHS System and GP and Pharmacist . . . . .	132
6.8	NHS System dependencies . . . . .	133
6.9	First iteration of resolution analysis . . . . .	135
6.10	Second iteration of resolution analysis . . . . .	137
6.11	Entailments analysis . . . . .	139
6.12	Entailments validation . . . . .	140
6.13	Trustworthiness requirement analysis . . . . .	141
6.14	Trustworthiness level before and after the trust analysis . . . . .	141
6.15	Survey approach . . . . .	145
6.16	Survey data collection methods . . . . .	150
6.17	Results related to JTrust modelling language . . . . .	152
6.18	Results related to JTrust tool . . . . .	153
6.19	Results related to JTrust methodology . . . . .	154

# Chapter 1

## Introduction

*...trust is a social good to be protected just as much as the air we  
breath or the water that we drink. When it is damaged, the  
community as a whole suffers; and when it is destroyed, societies  
falter and collapse*

---

Sissela Bok

Information Systems (ISs) exist in every aspect of our life and our society depends on them enormously. In particular, organisations need to process a rapidly growing amount of information and individuals rely on information systems for almost everything from health care and banking to the weekly shop at the supermarket. ISs have the ability to collect and store large volumes of information that can easily be accessed by anyone and from anywhere. Also, software, which is part of information systems, controls a vast number of systems such as e-health, e-government, and systems that exist in factories and systems that control air-traffic. Nevertheless, in the light of ambient, pervasive, and ubiquitous computing, the impact of information systems is still increasing significantly.

An information system is a collection of software, hardware, humans, and procedures, which aims to support the operations of an organisation. The combination of software and hardware provides a number of humans with information on specified topics of interest in a certain organisational context (Iivari and Hirschheim, 1996). So, an information system collects, stores, analyses, and extracts data information. Presently, information systems that were monolithic, isolated and independent have given their place to distributed information systems that operate on a worldwide scale, across open networks, and across different organisations.

Therefore, because of the importance and the complexity of information systems and the high demands placed on them, better software engineering approaches are required. Software engineering designs, implements, and deploys software for

information systems and on the other hand information systems and informing processes are used in software engineering, for example in requirements engineering and project management, in order to successfully embed software in socio-technical contexts. According to the definition given by IEEE (IEEE, 1990), "Software engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software". Software engineering has a number of phases (IEEE, 1990); first, the problem to be solved is analysed and the requirements are described in a very precise way. Secondly, a design is made based on these requirements. Finally, the construction process, i.e. the actual programming of the solution, is started.

The study of this thesis belongs to the first phase of software engineering, i.e. requirements engineering, because it addresses the challenges of developing trustworthy information systems that arise during the requirements definition. According to (Ross and Schoman, 1977), "Requirements definition is a careful assessment of the needs that a system is to fulfil. It must say why a system is needed, based on current or foreseen conditions, which may be internal operations or an external market. It must say what system features will serve and satisfy this context. And it must say how the system is to be constructed". The discipline that is concerned with requirements definition is called requirements engineering. According to (Zave, 1997) "Requirements engineering is the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behaviour, and to their evolution over time and across software families". The more careful we are during the requirements engineering phase, the larger is the chance that the ultimate system will meet expectations. However, requirements engineering involves modelling of the information system under development, which includes active components that have the choice of behaviour. So, by nature, there is uncertainty about whether the actual behaviour of components will match the expected behaviour, which is reflected in the requirement engineering models. This uncertainty requires the developer to decide whether or not she can trust active information system components to behave in an expected way, as described in her model, and build the system on such decisions that will determine its success. There is also uncertainty about the behaviour of the environment that has to be considered by the developer and reflected in the requirements engineering models, but this is outside of the scope of this thesis. The main contribution of this thesis therefore, is the methodology named JTrust. JTrust stands for Justifying Trust and the methodology aims to assist the developer in modelling and reasoning about trust relationships, model and analyse trustworthiness requirements, and as-

sess the trustworthiness of the system-to-be at a requirements stage, to eventually develop a trustworthy information system.

## **1.1 Motivation and problem statement**

Despite the reliance on information systems and the emergence of software engineering, these systems are often unreliable, prone to errors, and possess vulnerabilities that could be exploited in security attacks (Fortune and Peters, 2005). We are often faced with a choice between using a valuable (or even an essential) system, which is not fully trustworthy, or else forgoing the services it provides (Islam, Mouratidis, and Wagner, 2010). There is vast number of reported incidents about security and privacy breaches (Fortune and Peters, 2005) resulting in an especially big concern regarding security and privacy issues that prevent the full extent of their utilisation. Also, a lot of times the developed systems do not meet their functional requirements (Fortune and Peters, 2005). This leads to greater customer dissatisfaction and disappointment resulting in an abandoned system. The consequences of such failures can be devastating. They can be financial loses or loss of time or even loss of human lives.

Trust, therefore, is becoming an increasingly important issue for information systems that process and manage sensitive user and organisational information. As a result, there is an increasing interest in the development of trustworthy information systems. Systems that will be worthy of users' and other stakeholders' trust, that there is strong confidence that they will meet all their functional and non-functional requirements. To this end, new methods and tools for developers are required that will provide the know-how and guidance on developing trustworthy information systems.

The development of systems that are trustworthy will have many benefits. First of all, the successful adoption of information systems in society depends on whether these systems can be justifiably trusted by the users (Hasselbring and Reussner, 2006). So, trustworthy systems are more likely to be adopted by users. Also, by being trustworthy they will meet their functional and non-functional requirements, thus assuring customer satisfaction, which will lead to greater financial profits. Also, systems will be secure and protect human privacy, which is a major user concern.

In addition, being able to trust a software system is a prerequisite for its social acceptance (Cofta, 2007) and when there is trust in a software product it will increase its sales and the willingness of the users to pay even more for that product (Masthoff, 2007). Otherwise, the software system will be rejected and the development of that software system will result in a failure.



During the design developers know how to make a system that satisfies non-functional requirements, such as security and usability (Mouratidis and Cofta, 2010). Developers have also the knowledge and the tools to carry out the necessary tasks or they can even hire professionals who have expertise in a specific field. On the other hand though, they do not have adequate knowledge about trust, since trust is an interdisciplinary issue and requires knowledge from other sciences such as sociology, psychology, and social psychology and not adequate methods and tools have been developed yet. Alternatives for developers are trusted computing, trustworthy computing, and social trust. "Trusted Computing" entails that certain hardware components need to be trusted in order to provide security, while trustworthy computing promises security, privacy and reliability and hopes to gain trust of users. Social trust uses trust that is external to the system (Mouratidis and Cofta, 2010). Also, there is not a common understanding of trust among developers. Developing teams might consist of developers from different cultural backgrounds that each one of them possesses a different perception about trust (Mouratidis and Cofta, 2010). Trust is a complex notion and depends upon technical and non-technical issues of the social and organisational setting. Therefore, developers require trust related abstractions that can be used uniformly by all of them and across different projects.

In the past it was easier for a system be developed trustworthy, as it was very simple, isolated, and only depending on itself. Modern information systems though include not only technical components but also human components that exist in the environment of technical components. There is a distinction between the technical component, which is one or more computers that behave in a way to satisfy the requirements with the help of the software, and the environment, which is the part of the world with which the machine will interact and in which the effects of the machine will be observed. When the machine is put into its environment, it can influence that environment and be influenced by that environment only because they have some shared phenomena in common (Jackson, 1997). Nowadays technical components are very big in size and complex and interact with other components of the environment. Thus, the trustworthiness of the system depends also on other external technical and human components. Furthermore, modern information systems comprise socio-technical infrastructures that include large numbers of actors, including humans. Due to the need for constant interaction and communication with other systems and humans, which do not belong to their infrastructure, technical components need to interact with systems and humans that they might not have interacted before. In fact, they might depend on other systems and/or humans to accomplish tasks and operations that directly affect their operation. Consider for example the scenario where an information system depends on another system

for information that is crucial for completing some of its operations. In such scenario, trust, to both humans and other systems, is an important issue for modern information systems as they depend on entities (humans and systems), over which they do not have direct control, for resources to achieve their goals. It is therefore important, in order to understand the risks involved in such dependencies, to understand the various trust relationships that an information system might be part of (Pavlidis, Mouratidis, and Islam, 2012; Pavlidis et al., 2012; Zarrabi et al., 2012; Pavlidis et al., 2014).

The developer's perspective needs to be considered in the analysis of the system under development. It is the developer who is assigning new goals and tasks to the components of the information system and her view of the system is important to the whole trustworthiness of the system.

In addition, to establish systems trustworthiness, it is important that trust relationships between the system and other entities and trust assumptions, which are usually made during the development process, are properly identified and analysed. Therefore, it is important in order to understand the consequences that trust relationships might have on the operation of an information system to be able to analyse, in a systematic and structured way, the various trust assumptions that are usually made during the development process of information systems (Cofta, Laco  e, and Hodgson, 2010; Cofta, 2007). By trust assumptions, we refer to assumptions that are made by developers and/or stakeholders related to the various trust relationships that a system is part of. The assumptions are underlying the analysis of the system and can undermine its trustworthiness.

The main measure of success of a software system is the degree to which it meets its purpose. Thus developers have to make sure that the software-to-be will meet its purpose. Currently, there is a lack of techniques that attempt to measure the trustworthiness of the system under development at an early stage.

The introduction of an information system causes not only technical changes but also social changes. When introduced inside a social environment the social agents need or are required to change their behaviour accordingly. Sometimes humans resist and do not accept the changes. By overlooking whether the humans will accept the changes and behave in an expected way, it results in systems that are not trustworthy and not socially accepted.

Employing control mechanisms to solve problems increases the complexity and cost of a system. On the other hand though, employing trust leads to systems that are less costly and less complex. However, there is a trade-off between trust and control. Relying on trust entails the acceptance of risk in order to get the benefits and relying on control you spend more resources in order to increase the assurance.

Although the literature provides a large body of work related to engineering of trust in information systems, it fails to provide evidence of a systematic and structured way to model and reason about trust relationships during the early stages of the development process and identify trustworthiness requirements. It is important for such analysis to be made during the early stages since at that stage of the process is when the trust relationships are formed and therefore it is where changes need to take place if the relevant assumptions about trust relationships do not hold.

## 1.2 Research questions

We summarise the following research questions that, as usual in software engineering (Wieringa and Heerkens, 2006), are a mix of knowledge, design, and empirical questions:

- RQ1: What are the required trust abstractions and their relationships in order to reason about trust relationships at a requirements stage?
- RQ2: What are the required abstractions and their relationships that can ensure the development of trustworthy information systems at a requirements stage?
- RQ3: How can we assess trustworthiness of the system under development at a requirements stage?
- RQ4: How well does the methodology support modelling and reasoning about trust relationships?
- RQ5: How well does the methodology support trustworthiness requirement modelling and analysis?
- RQ6: How well does the methodology assess the system trustworthiness at a requirements level?

## 1.3 Research aims and objectives

The aim is to develop a novel methodology to allow modelling of, and reasoning about, trust relationships in a structured and coherent way. By trust relationship we mean the relationship between the developer and the components of the information system that are modelled in her requirements model. The trust relationships of the developer are eventually becoming trust relationships of the system-to-be

once the system is put in operation. In addition, the goal is also a methodology to allow modelling and reasoning of trustworthiness requirements that ensure the trustworthiness of the system-to-be. Furthermore, we aim to develop a methodology to allow the assessment of the trustworthiness of the system under development at a requirements stage.

To achieve this aim the following objectives have been defined:

- Develop a modelling language for capturing and modelling relationships of trust at a requirements stage.
- Develop methods and techniques to analyse and reason about trust relationships and identify trustworthiness requirements from the early stages of the development, and integrate such methods and techniques to form a methodology.
- Provide techniques to assist the automatic evaluation of the system trustworthiness.
- Assess the developed methodology by applying it to a concrete and complex real world case study and receive experts' views through questionnaires.

## 1.4 Research contributions

Besides successfully addressing the research problem, this research has a number of novel contributions to the state of the art and knowledge. These four novel contributions along with the development of a CASE tool are introduced subsequently:

1. **The first contribution of the thesis is the analysis of the problem of establishing information systems trustworthiness.** This included the identification of problems, limitations, and challenges of the state of the art with respect to trustworthy information system development, and more particularly reasoning about trust relationships and identifying trustworthiness requirements.
2. **Definition of trust abstractions for describing, understanding, and analysing the complex notion of trust.** The definition of the trust abstractions along with their relationships with existing Goal Oriented Requirements Engineering (GORE) abstractions and control abstractions is a part the modelling language of the proposed methodology. It establishes a common understanding of trust among developers within a technical setting of a project that might have a different cultural background and possess different views

about trust. They enable the reasoning about the trust relationships that exist with the system environment. By using the trust-based concepts such as resolution, developers can show explicitly why there is trust in a dependency. The use of the concept of reported trust and control create new dependencies that reveal the indirect trust relationships. Through this way direct and, in particular, indirect trust relationships become explicit. The trust abstractions enable the identification and reasoning of trust relationships that leads to the natural surface of trust assumptions that need to be examined in terms of their validity.

3. **Definition of control abstractions for describing, understanding, and analysing the complex notion of control.** The definition of control abstractions along with their relationships with the trust abstractions and existing GORE abstractions is part of the proposed modelling language. The control abstractions enable the modelling and analysis of trustworthiness requirements. Control abstraction such as observation allows the specification of the functionality that is required in order to monitor a specific resource in order to verify if a system component is behaving in the expected way. Moreover, control abstraction such as deterrence allows the specification of functionality that prevents the achievement of a system component's own goal in order to compel to behave in a specific way.
4. **Development of a methodological process that employs the modelling language for the development of trustworthy information systems.** The proposed process is systematic and structured with defined activities and tasks that the developers can easily follow and will guide them towards the development of trustworthy information systems. It allows the incremental development of a trust model of the system under development. It starts with the identification and reasoning of trust relationships and the early consideration of trust relationships enables to identify potential vulnerabilities to the system trustworthiness. In particular potential vulnerabilities to the system trustworthiness are identified and they are resolved through trust or control means. Then it includes the modelling and analysis of trustworthiness requirements. The control means represent the trustworthiness requirements that fill in the gaps in the chain of trust relationships. In addition, in this manner the trade-off between trust and control is becoming more explicit for the developer and assist him to take better decisions by knowing what exactly is the situation and potential implications. Therefore, the developer can avoid unnecessary control functions, which can increase cost, complexity and time to

delivery. Also, the methodology can be used not only for constituting a technical system trustworthy, but also on making the whole socio-technical system trustworthy. By applying the methodology not only from the perspective of the technical system but also from the perspective of all entities in the system environment then the information system trustworthiness can be established. Moreover, there is contribution towards the assessment of the trustworthiness of the system. The process includes algorithms that enable the evaluation of whether the system will achieve its goals by considering the goals that are assigned to it and also goals that are assigned to components of the information system and are expected to be accomplished. Thus, trustworthiness is examined from a holistic perspective in order to capture all the properties that are important in developing trust and not only a subset.

5. **Development of a CASE tool for supporting the methodology.** The tool enables the construction of the trust model and the automatic assessment of the system trustworthiness by executing the proposed algorithms.

## 1.5 Research approach

The study started with the identification of the research problem, which was to investigate the issues involved in developing trustworthy information systems and networks and develop a novel methodology to allow modelling and reasoning of trust issues in a structured and coherent way. An extensive literature review was carried out which included a comparison framework for the evaluation of related work in a consistent way and not in an ad-hoc way. Then the research questions were identified that drove the rest of the process and in particular the identification of the research aims and objectives. Figure 1.1 depicts the research approach of this thesis.

After the research questions have been identified a research method has to be selected. The main research methods for software engineering are the deductive and inductive approaches (Partridge, 1997). The deductive method is reasoning from the general to the specific, while the inductive method is reasoning from the specific to the general (Partridge, 1997). Also, in the deductive method we have an abstract generalisation for the specification of the problem, while in inductive method the problem is satisfied in terms of behaviours, because you know more about specific behaviours than about the problem abstraction (Partridge, 1997).

Deductive reasoning moves from the general to the particular. It takes a general premise and deduces particular conclusions. A valid deductive argument is one

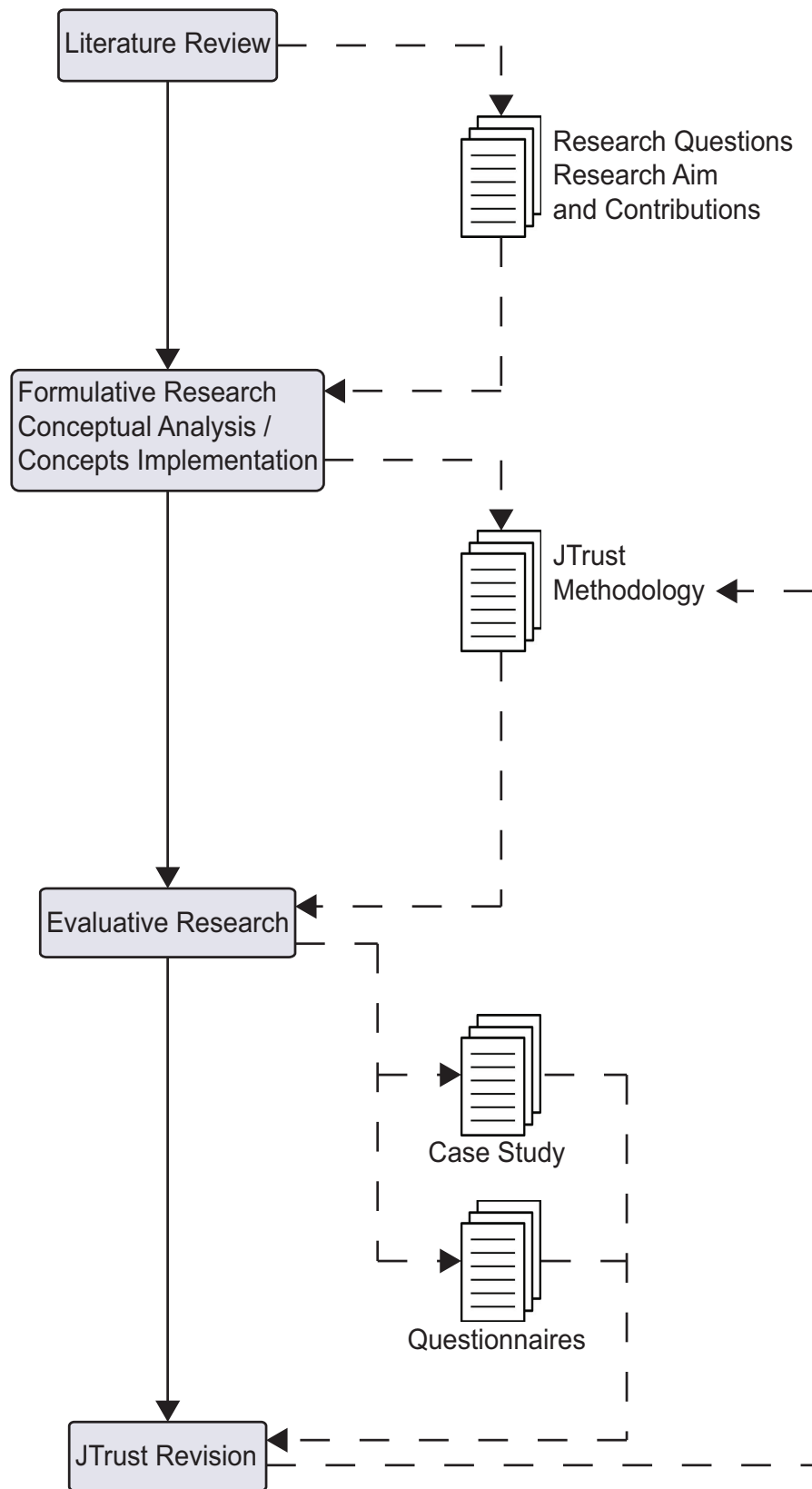


Figure 1.1: Research approach

in which the conclusion necessarily follows from the premise. On the other hand inductive reasoning moves from the particular to the general. It gathers together particular observations in the form of premises, and then it reasons from these particular premises to a general conclusion. In our case, since we have some initial goals and objectives, the deductive is more preferable because the inductive method cannot guarantee correct results in the sense that they meet the requirements specification (Michie, 1982), in our case the initial goals and objectives. On the other hand, the deductive approach provides adequate guarantees. However, for the individual components and requisites of the methodology the inductive approach will be followed. In contrast with the deductive approach, which limits the scope of the results, the inductive approach does not limit the scope of the results. In addition, humans have difficulty in expressing in systematic terms the rules of their expertise, but they are good at taking decisions on specific cases (Michie, 1982).

A hypothesis in software engineering science is a description of the new object to be constructed, which in our case was the new development methodology. Therefore, the hypothesis was a specification of requisites of the new object to be constructed (Marcos, 2005), which was the proposed methodology. For the solution an analysis of similar cases and a process of imagination and creativity was carried out and for the verification the application of the methodology on a prototype case study (Marcos, 2005).

Various validation techniques exist and the most common are (Shaw, 2003):

- Analysis. The results are analysed and have been found satisfactory.
- Evaluation. Given the stated criteria the results meet the criteria.
- Experience/Case study. The results have been used on real examples and there is evidence of their correctness or usefulness or effectiveness.
- Example. A demonstration of the findings.
- Persuasion. Validation by persuasion is rarely sufficient. It is sufficient only in some cases of feasibility research questions.

For this research project a combination of analysis, evaluation and experience/case study were used as validation techniques.

## 1.6 Publications

Parts of the presented research have been published in journals, conferences, and workshops.



- Michalis Pavlidis, Shareeful Islam, Haralambos Mouratidis, Paul Kearney. : Modelling Trust Relationships for Developing Trustworthy Information Systems. *International Journal of Information Systems Modelling and Design (IJISMD)* 5(1), (2014).
- Michalis Pavlidis, Haralambos Mouratidis, Shareeful Islam. : Modelling Security Using Trust Based Concepts. *International Journal of Secure Software Engineering (IJSSE)*, Volume 3, Issue 2, (2012).
- Michalis Pavlidis, Evangelia Kavakli, Philemon Bantimaroudis, Haralambos Mouratidis, Christos Kalloniatis, Stefanos Gritzalis. : The Role of Trust for the Development of Cultural Internet-Based Systems. 10th European, Mediterranean and Middle Eastern Conference on Information Systems (EM-CIS), Windsor, United Kingdom, October (2013). [Best Theoretical Paper Award]
- Michalis Pavlidis, Haralambos Mouratidis, Christos Kalloniatis, Shareeful Islam, Stefanos Gritzalis. : Trustworthy Selection of Cloud Providers Based on Security and Privacy Requirements: Justifying Trust Assumptions. 10th International Conference on Trust, Privacy and Security in Digital Business (TrustBus), Prague, Czech Republic, August (2013).
- Fatemeh Zarrabi, Michalis Pavlidis, Haralambos Mouratidis, Shareeful Islam, David Preston. : A Meta-model for Legal Compliance and Trustworthiness of Information Systems. *Advanced Information Systems Engineering Workshops, Lecture Notes in Business Information Processing, Springer, Volume 112*, 46-60, (2012).
- Michalis Pavlidis, Haralambos Mouratidis, Shareeful Islam, Paul Kearney. : Dealing with Trust and Control: A Meta-Model for Trustworthy Information Systems Development. *International Conference on Research Challenges in Information Science (RCIS)*, IEEE, (2012).
- Michalis Pavlidis, Shareeful Islam, Haralambos Mouratidis. : A CASE Tool to Support Automated Modelling and Analysis of Security Requirements, Based on Secure Tropos. *Lecture Notes in Business Information Processing, Springer, Volume 107*, 95-109, (2012).
- Michalis Pavlidis. : *Designing for Trust*. CAiSE Doctoral Consortium, London, United Kingdom, June (2011).

- Michalis Pavlidis, Shareeful Islam. : SecTro: A CASE Tool for Modelling Security in Requirements Engineering using Secure Tropos. CAiSE Forum London, United Kingdom, June, (2011).

## 1.7 Structure of the thesis

The structure of the rest of the thesis is as follows. Part I describes the state of the art and consists of Chapter 2, which provides background information about trust and trustworthiness. It is explained how trust changes over time, and how important is the context for the decision to trust or not to trust. Also, it defines the meaning of system trustworthiness in the context of the current work. This chapter also defines a set of requirements that are essential for software engineering methodologies and Computer Aided Software Engineering (CASE) tools. Moreover, it presents the related work in the field of trust engineering. It offers a classification of the existing related work into trust modelling, trust management, security engineering, goal satisfaction reasoning, human computer interaction, trusted computing and computational trust.

Part II is the main contribution of the thesis, which presents the JTrust methodology and consists of three chapters. Chapter 3 describes the requirements of the methodology and its structure. Also, it introduces a running example that is used throughout the thesis. In addition, it describes the JTrust modelling language proposed by this research. The concepts of the language are presented along with the meta-model of the language, which shows the relationships between the concepts. Also, the trustworthiness requirements are described in this chapter and the formulas for calculating the system trustworthiness are presented. Moreover, The content of this section was published at the IEEE Conference of Research Challenges in Information Science and at the International Journal of Secure Software Engineering.

Chapter 4 describes the process of the proposed methodology. In particular, the activities included in the process are described. These are, the identification of the actors and their dependencies, the identification of resolutions, the identification of entailments, and the identification of the trustworthiness requirements. In addition, the activity of the assessment of system trustworthiness is presented towards the end of the chapter. The content of this chapter was published at the International Journal of Information Systems Modelling and Design.

Chapter 5 describes the JTrust CASE tool that was developed to support the activities of the methodology. Particularly, it explains how the tool supports the developer and what are the benefits and the automations that it is offering.

The third and last part of the thesis presents the evaluation of the research and

conclusions. In particular, Chapter 6 demonstrates the validity of this research. This is demonstrated in two ways. First, through the application of the methodology in real case study. And secondly, by getting feedback from experts through the use of questionnaires.

Chapter 7 provides a critical discussion regarding the proposed trust engineering methodology and it concludes the thesis. It discusses the contributions and the significance of this research and it describes directions for future work.

Figure 1.2 provides a diagrammatic model of the thesis' main claim, and the propositions upon which this is based. For the thesis to hold we claim that each of these concepts or statements are valid. The arrows between the boxes indicate that the tail concept acts as ground for the head.

The validation of the thesis is based on the validity of the case study and the survey. The validity of the case study and the survey is contingent on the soundness of the modelling language, the methodological process, the case study methodology used to design the case study, the survey methodology used to design the survey, and the tool used to support the proposed methodology. The figure illustrates how the motivation for developing trustworthy information systems also motivated the design of the literature review. Given the broadness of the notion of trust and how it was dealt from different perspectives was used to drive this review. Based on this review limitations were identified in Chapter 2. These were the lack of appropriate abstractions and constructive techniques that were required to reason about trust and its complementary notion of control. Also, the review helped to devise the methodology for developing the proposed approach and validate it. Guided by the research methodology, in Chapter 3 the JTrust meta-model was developed, which provided the foundation of the modelling language. In Chapter 4 a methodological process was developed based on modelling language. The modelling language and the process suggested design principles that the tool support should include and specific characteristics were derived. These characteristics informed the architecture and the functionality of the JTrust tool prototype presented in Chapter 5. The JTrust methodology was validated using the case study and the survey described in Chapter 6.

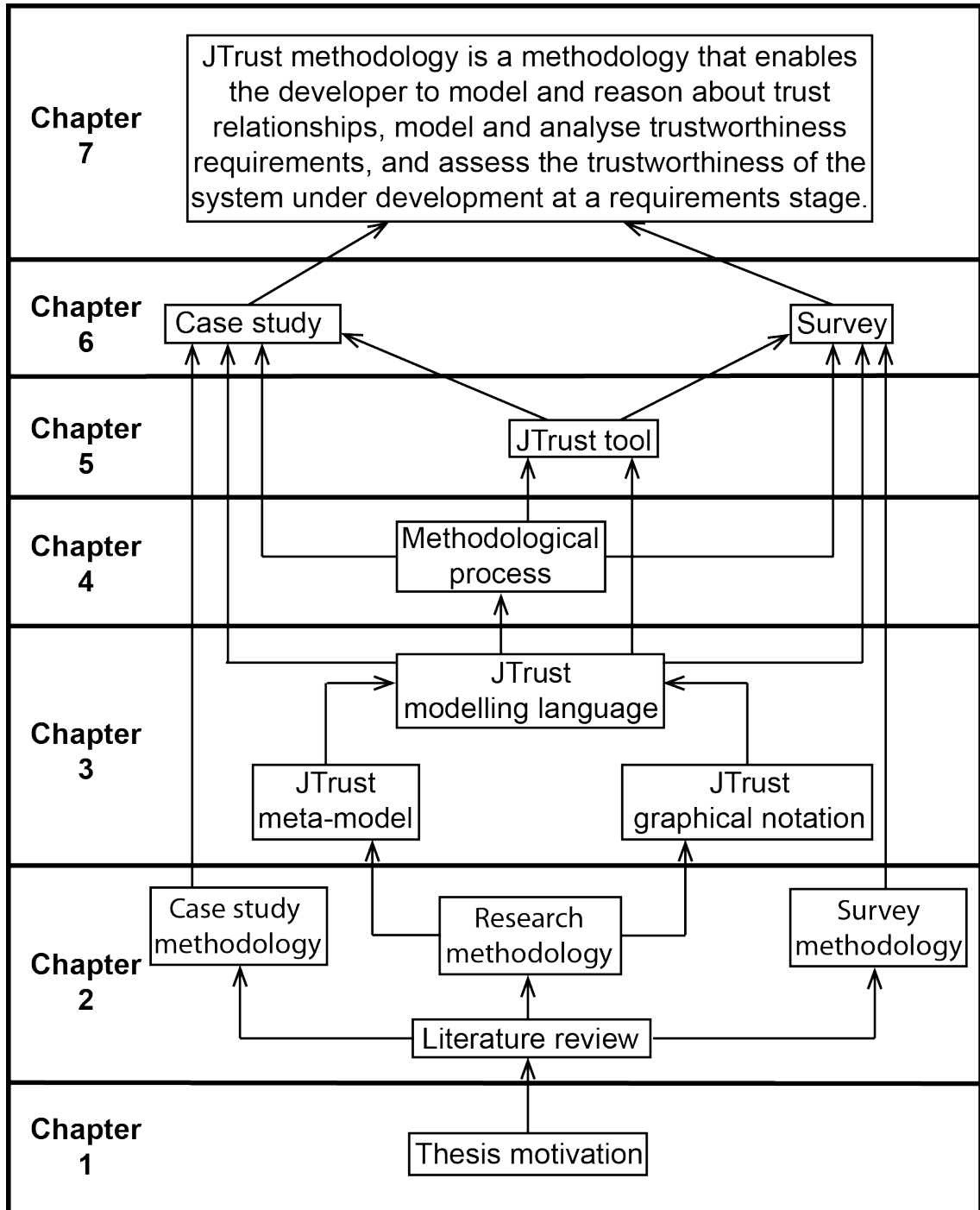


Figure 1.2: Thesis claim and supporting propositions

## Part I

# State of the Art

## Chapter 2

# Literature review

In this chapter, we review the current state of the art in the area of trustworthy information systems development, which is broad, multidisciplinary, and includes diverse approaches belonging to different research sub-areas. A core element in this area is trust. This chapter is divided in two parts. Sections 2.1, 2.2, 2.3, 2.4, and 2.5 belong in the first part where we provide background information about trust and we discuss definitions of trust, In addition, in this part we review the literature that is relevant and defines our research baseline, covering trust and trustworthiness in the context of information systems in general and in the context of this thesis more specifically, development methods and CASE tools for information systems, and evaluation methods for software engineering research. The second part of this chapter is the section 2.6, which provides a comprehensive survey on approaches to trustworthy information systems development, including their main contributions and criticism. A core element in these works is trust, so we look into how these approaches are dealing with trust. The areas that are relevant to our proposal are trust modelling during requirements analysis which deals with capturing the rationale of trust decisions, trust management, security engineering that considers trust in order to make the system more secure and in effect more trustworthy, goal satisfaction reasoning techniques that can provide confidence that the system can achieve the goals that has been assigned, Human Computer Interaction that deals with methods of gaining user's trust but focusing mostly at a user interface level, Trusted Computing that ensures the trustworthiness by using a hardware component as a controller for the rest for the components, and computational trust that develops models that can be used by artificial agents when required to make a trust decision.

## 2.1 Trust and trustworthiness

Trust has been the object of research for many years. Here we outline several definitions of trust and then we make an observation about the common characteristics of trust:

- According to *Oxford Dictionary*, "trust is the firm belief in the reliability or truth or strength of an entity". While, the Webster dictionary defines trust as "a confidence dependence on the character, ability, strength, or truth of someone or something". However, both of these definitions give a very general view of trust and enable many interpretations.
- A definition given by Deutsch (1962) is "a) an individual is confronted with an ambiguous path, a path that can lead to an event perceived to be beneficial or to an event perceived to be harmful; b) he perceives that the occurrence of these events is contingent on the behaviour of another person; and c) he perceives the strength of a harmful event to be greater than the strength of a beneficial event. If he chooses to take an ambiguous path with such properties, he makes a trusting choice; else he makes a distrustful choice".
- Luhmann (1979) defined trust as "a means for reducing the complexity of society; complexity created by interacting individuals with different perceptions and goals".
- Barber (1983) defined trust as the subjective expectation of future performance.
- In addition, Gambetta (1988) has defined trust as "a particular level of the subjective probability with which an agent assesses that another agent or group of agents will perform a particular action, both before he can monitor such action (or independently of his capacity ever to be able to monitor it) and in a context in which it affects his own action".
- Mayer, Davis, and Schoorman (1995) defined trust as "the willingness of a party to be vulnerable to the actions of another party based on the expectation that the other party will perform a particular action important to the trustor, irrespective of the ability to monitor or control that other party".
- Rousseau et al. (1998) defined trust as "a psychological state comprising the intention to accept vulnerability based upon positive expectations of the intentions or behaviour of another".

The meaning of trust given by researchers usually reflected their science. In psychology trust represents a personal attribute (McKnight and Chervany, 1996), which is developed through the early psychological development of an individual and is conceptualised as a belief, expectancy, or feeling that is deeply rooted in the personality of the individual (Lewicki and Bunker, 1996). Therefore any attempt to investigate the development, maintenance and stability of trust should consider the individual differences, which are a result of different past experiences.

In social psychology trust is a phenomenon that occurs in a social relationship (McKnight and Chervany, 1996) and is based on the expectation of the other party in the relationship (Kini and Choobineh, 1998). Trust is also expectancy in the occurrence of future events and is strongly related to the subjective probability that the individual assigns to the occurrence of these events (Rempel, Holmes, and Zanna, 1985). With respect to the decision to trust it involves the importance of the issue to the individual (Kini and Choobineh, 1998) and, according to McKnight and Chervany (1996), the attributes of the individual upon which another individual will base his decision to trust or not are different from situation to situation and from relationship to relationship.

In sociology trust is viewed as something normal in nature (McKnight and Chervany, 1996) and as a social reality that is functionally necessary for the continuance of harmonious social relationships (Lewis and Weigert, 1985). The existence of trust is that it increases engagements and activities in a society. While on the other hand, lack of confidence will lead to no participation and lack of trust will lead to no activities (Luhmann, 1988). For example, citizens show trust in their governments, patients in their doctors, clients in the lawyers and students in their teachers. When trust is absent in such cases, then crises and riots emerge. Furthermore, (Luhmann, 1988) states that trust exist in order to represent a connection between the familiar and unfamiliar of our world.

In economics trust is seen as a cause that reduces opportunism in a transaction and as a consequence it can lead to lower transaction cost for the participants (Rousseau et al., 1998). If there is trust, negotiations are easier and shorter and there is less need to monitor and enforce the agreement. In other words, lack of trust is an obstacle to personal relationships and conducting businesses. In addition, trust allows the reduction of complexity (Luhmann, 1988) by decreasing the control mechanisms, and as a result there are savings in valuable resources such as time and money. Also, during crisis and uncertainty trust becomes more important and proves to be a valuable asset (McKnight and Chervany, 1996). When used properly, trust is enabler of building collaborations among the participating actors, a necessary antecedent for cooperation (Axelrod, 1984). However, when abused, trust can act



as a stopping block for successfully achieving a goal. Moreover, trust in economics is considered to be a rational choice mechanism (McKnight and Chervany, 1996) but, nevertheless, the decision to trust or not can be affected both by cognitive and emotional elements (McAllister, 1995). The cognitive element refers to a rational assessment of risk, the other party's reliability and competence, and is therefore more task-oriented. On the other hand, the emotional element refers to attraction, in the short term, and loyalty, in the long term. Its orientation is therefore more inter-personal (Egger, 2003).

Although, there is variety of definitions of trust, there are some common characteristics identified in most of the definitions, such as:

- **Interaction:** Trust entails the existing or potential interaction between two parties. We can say that the two parties are engaged in a trust relationship.
- **Expectation:** When there is trust it means that there is an expectancy of one party from the other party, or that both parties have expectations from the other party. When a party has an expectation he is the trustor and the other party is the trustee. Therefore, in a trust relationship a party can be simultaneously a trustor and a trustee.
- **Uncertainty:** Trust acts as a means to remove the uncertainty about the outcome of an interaction between the trustor and the trustee. In other words, we consider trust as a means of creating confidence in the outcome of an interaction. In society the problems of confidence are often solved by control mechanisms such as the law and politics (Luhmann88). For example, an individual has confidence that his right to express himself will not be taken by another individual because of the law. On the other hand, the case of trust is different.
- **Vulnerability:** When one party trusts another party then it becomes vulnerable to a potential negative outcome. The implications of the negative outcome are the risk that the trustor is willing to take. Trust therefore includes a decision or action and consequently risk. Risk is like a mechanism to represent the difference of the controllable and the uncontrollable (Luhmann, 1988). Nobody can predict the future and even if something is planned to bring certain result, it might lead to a different result, as there will always be parameters that cannot be fully controlled. Therefore, the possibility of different results than the ones planed is named risk, which is a term that represents that unwanted results might be the consequences of our actions (Luhmann, 1988).

- Decision: Trust entails a decision about choosing to take or not take an action. It can lead to disappointment, if you take the action and neglect the possible bad outcome. The reaction to a bad outcome is internal in the case of trust since you can blame yourself and regret for taking the action (Luhmann, 1988).
- Trustor subjectivity: There is always subjectivity in case of trust. The decision to trust or not trust depends on the perspective of the trustor and his personal characteristics and past experience.
- Trustee attributes: The decision to trust or not trust depends on the attributes of the trustee as well. Therefore, an individual may trust another individual who has the desired attributes even though the trustor individual is inclined not to trust due to his personal characteristics or past experience.
- Context: The same level of trust is not always developed between certain parties. The conditions of the outside world affect the development of trust (Luhmann, 1988), so different level of trust, or no trust at all, will be developed between two parties if they happen to be in different environment conditions. If we trust a doctor for suggesting a medicine, it does not mean that we have to trust her when she is suggesting a specific meal at a restaurant. The reputation as a good doctor it does not help if we are looking for a plumber.

Based on the aforementioned observations, we have adopted the following definition for use throughout this thesis. This definition considers the aforementioned characteristics as it is not too narrow to leave important concepts outside and it is broad enough to capture the richness of the concept of trust.

*Trust is positive expectations of the behaviour of another party from whom he might be positively or negatively affected (Möllering, 2005).*

Having defined trust in the context of this research, we need to understand the process of trusting. Trust is not static, but dynamic, which means that it is not stable during a time period but it changes over time. When two parties engage into a relationship there are two categories of factors that have an impact on the trust process, the extrinsic and intrinsic trust factors (Jøsang, Keser, and Dimitrakos, 2005). The extrinsic trust factors are all the information about the trustee that is collected by the trustor without any direct experience, such as the trustee's reputation. On the contrary, the intrinsic trust factors are the information that the trustor collects about the trustee while having a direct experience. In the early stages of the trust relationship the extrinsic trust factors have a greater

impact in the trustor's trust decision. Almost every relationship begins with an initial phase where the extrinsic factors are more important (McKnight, Cummings, and Chervany, 1998), however, as time goes by the intrinsic trust factors become more important since the trustor can make his decision on information that has been collected from direct experience. Of course, it has to be mentioned that this process does not consider how the trustor due to his personal subjectivity interprets the intrinsic and extrinsic factors. In other words two different individuals can take a different trust decision, even though the intrinsic and extrinsic factors are the same.

The optimum condition though in the trust relationship between individuals that will benefit the society is reached not when they show unlimited and un conditional trust but when there is justified trust (Braynov, 2002). Trust that is justified is the key. Justified trust is when trust of the trustor matches the trustworthiness of the trustee, and as a result the maximum benefit occurs (Figure 2.1). Trusting less is a loss of opportunities, while on the contrary trusting too much makes you vulnerable (Cofta, 2007; Cofta, 2008). For example, let us consider the case of a software company, where its software developers can develop a software application in one year. When a client asks for a software application the management of the company has to make a decision and agree with the client on delivery date. If the company management under-trusts its software developers and offers a time more than a year, then it risks losing the contract to a competitor. On the other hand, if the company management over-trusts its software developers and commits to a time of less than year, then it becomes vulnerable as the product may not be ready in time and there may be financial consequences. The key is for the company management to assess the trustworthiness of its software developers with respect to their ability of developing the software application and based on this assessment show justified trust.

Trustworthiness is a characteristic of an individual or thing that is the object of that individual's trust. If the object of our trust is worthy of that trust, then it will fulfil our expectations and our trust will be rewarded, not betrayed.

In (Cofta, 2007) the characteristics that express the trustworthiness of an actor can relate to continuity, competence, or motivation. Continuity means that the current relationship between the two actors will continue in the future and it will not be only temporary. Competence refers to the capability of the trustee to support the trustor. An example of competence evidence in a student teacher relationship is if the teacher has a teaching certificate. Motivation is whatever drives the trustee to support the needs of the trustor and an obvious example is if the trustor's interest matches with the trustee's interest. In addition, McKnight and Chervany (2000) define the concepts of competence, benevolence, integrity, and predictability regarding

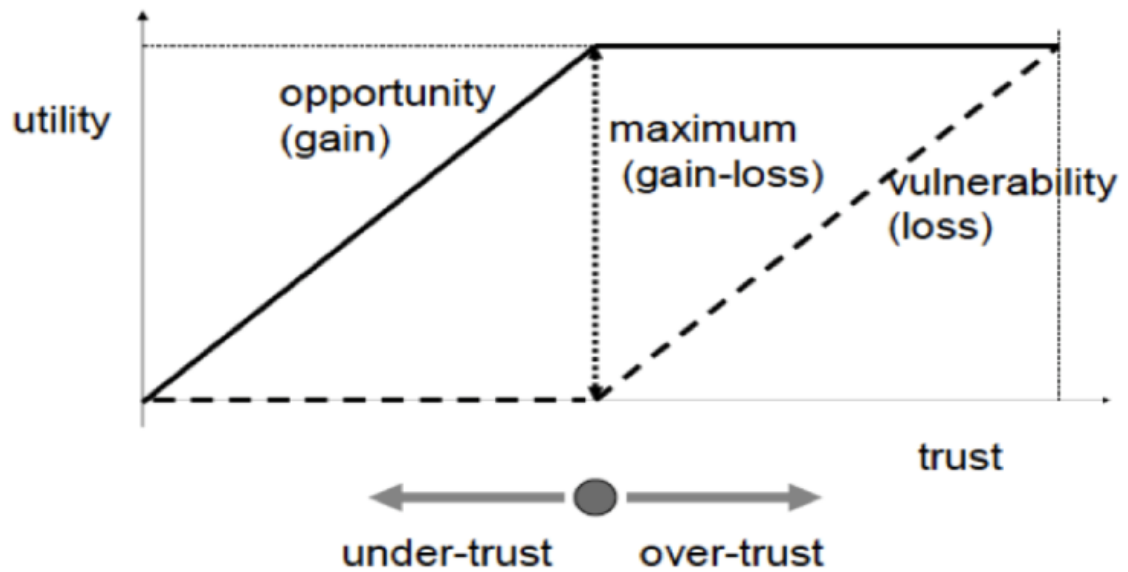


Figure 2.1: Optimum trust level

trustworthiness. Competence means that the trustee has the power or the ability to do what it is needed by the trustor, while benevolence means that the trustee also cares about the trustor and is motivated to act in his interest. Integrity means that the trustee is honest and he will keep his promise to the trustor and he will fulfil the agreement, while predictability means that the trustee's behaviour is consistent enough so that the trustor can forecast his future behaviour. Furthermore, in (Riegelsberger, Sasse, and McCarthy, 2005) the properties regarding trustworthiness are called intrinsic properties that consist of ability, internalized norms, and benevolence. Ability is all the skills, competences, and characteristics that are required by the trustee in order to deliver in his relationship with the trustor. Internalized norms are all the principles that are considered acceptable by trustee and he behaves according to them. Finally, benevolence is when the trustee is in a relationship with a trustor as part of his own gratification and in such a relationship the trustee does not expect any return from the trustor.

Our conclusion is that the competence or ability of the trustee to do what is expected from the trustor is the basic characteristic of trustworthiness as it appears in all the definitions given by the researchers. Other characteristics such as benevolence and motivation may be important as well but our focus will be concentrated on the competence characteristic once we transfer to the context of information systems. Competence depends on the specific trust relationships and its context. There is therefore trust between the trustor and the trustee when the trustee possesses enough evidence of competence that are considered signs of trustworthiness

within the trustor's social context. The more such characteristics the trustee possesses the greater the trust of the trustor in the trustee will be.

## **2.2 Information System trust and trustworthiness**

Having defined trust and trustworthiness in general, can we trust specifically technical entities, such as computers, mobile devices, or information systems for our day-to-day activities? Do we actually trust the system, the humans who have developed the system or the humans who operate the system? The attribution of trust is still an on-going discussion. On one hand we have researchers (Friedman, Khan Jr, and Howe, 2000; Shneiderman, 2000) who argue that trust is attributed only to humans as they possess an intentional behaviour and free will and they can behave in unpredictable ways. And on the other hand, there are researchers who claim that trust can be attributed to technical systems as well. Kini and Choobineh (1998) consider trust in the Internet as a form of social trust that is required, since users are not technically capable of understanding the technology of the Internet, but they are willing to use it. Miller and Voas (2009) discuss about what is trust in software systems. They claim that trust is a relationship between people or between people and a thing and to trust someone or something is to act as if the object of trust will perform as promised or as required. As software artefacts become increasingly sophisticated, trust in them becomes increasingly similar to trust in humans. That is, when software artefacts exhibit behaviours similar to humans, the relationship of trust to such artefacts will resemble the trust humans establish with other humans.

We subscribe here to the latter stance with respect to information systems. Information systems consist not only of technical components but also human components that have intentional behaviour and collaborate with the technical components for the fulfilment of goals. A natural consequence is that information systems inherit their human component's intentional behaviour and can be characterised as intentional entities. Moreover, today's information systems have become so great in size and complexity that it is difficult for users to predict their behaviour. Out of this difficulty arises the need of trust because trust is about future expectations. In that sense we argue that trust can be attributed to information systems as well.

In the literature there are a lot of definitions given for trustworthy information systems. Miller and Voas (2009) define trustworthiness as a characteristic of a person or a thing that is the object of someone's trust. It will fulfil our expectations and our trust will be rewarded. If a person is trustworthy, it is considered a virtue, so if a software artefact is trustworthy, then it is considered a mark of high quality. Berzins (2004) argues that in an ideal world, trustworthy systems would carry

absolute guarantees that the software will perform its required functions under all possible circumstances, will do so on time and will never perform any actions that have hazardous consequences. According to Jayaswal and Patton (2006), a trustworthy system is a system that has the capability of meeting customer trust and developing the capability to meet their stated, unstated, and even unanticipated needs. Moreover, Avižienis, Laprie, and Randell (2004) claims that trustworthiness of a software system is the assurance that the system will perform as expected.

Based on the previous definitions we observed that trustworthiness of an information system is a concern for the parties involved in the development and operation of the system. In addition, the involved parties expect from a trustworthy system nothing less than meeting their expectations. Therefore, throughout this thesis we have given the following definition for a trustworthy information system:

*Trustworthy information system is an information system that meets all the positive expectations of the stakeholders.*

In the literature there has been extensive research to define the components of system trustworthiness. According to Schneider, Bellovin, and Inouye (1999) trustworthiness is a holistic property, encompassing security, correctness, reliability, privacy, safety, and survivability and it is not sufficient to address only some of these diverse dimensions. Hoffman, Lawson-Jenkins, and Blum (2006) proposed and extended a trust model, which considers privacy, security, reliability, usability, safety, availability, and user expectations as subcomponents of trust.

All these issues are components of system trustworthiness as they are stakeholder expectations that are widely considered signs of trustworthiness and trustworthy systems must do what stakeholders expect and not something else, despite environmental disruption, human user and operator errors, attacks by hostile parties, and system design and implementation errors. Also, trustworthy systems must be able to produce reliable and authentic information. More particularly:

- Privacy of personal information plays a very important role in building trust in an information system. Today, information systems have the ability to collect and store personal information very easily and providing wider access. So, there is an increased risk for the personal information to be intentionally or unintentionally disclosed and will result in decreasing users' trust in the information system (Rohm and Milne, 2004). There are multiple examples in e-commerce and e-banking where the trustee is required to maintain the privacy of the customer's name, address and credit card details.

- Regarding security, if we do not consider trust in the design, then we might end up with a system that has security measures that are not needed and that they just make the collaboration of the users through the software system and with the system more difficult. On the other hand, security measures might not be taken in cases where it is assumed there is trust among the users or the system when it actually there in no such trust (Giorgini et al., 2004). Rasmusson and Jansson (1996) define two types of security, hard and soft security. Hard security is all the security mechanisms that protect the systems against any potential attack. However, there are cases that we do not only want to prevent an attack to a system resource but to protect ourselves from a provider of a harmful or low quality resource and the approach of protection in these cases is called soft security. Therefore, the receiver of the resource needs to show trust only to those resource providers that are trustworthy. Security is the concurrent existence of availability, confidentiality, and integrity (Avizienis, Laprie, and Randell, 2004).
  - Availability when the service of the software system is always ready for the user to use.
  - Confidentiality means that there is no disclosure of information to unauthorized users.
  - Integrity is when the information is not improperly altered.
- Reliability of a system can be defined as "the probability that a system will perform a specified function within prescribed limits, under given environmental conditions, for a specified time" (Stapelberg, 2009). Also, Avizienis, Laprie, and Randell (2004) define system reliability as the continuity of providing the correct service. The attribute of reliability of a system as a trustee contributes significantly to its trustworthiness. When a trustor assigns a subjective probability to the behaviour of the trustee this is called reliability trust and it excludes situational parameters (Jøsang, Ismail, and Boyd, 2007).
- Another additional attribute of trustworthiness is usability. According to Nielsen (1994) and Shackel (1991) usability is:
  - How easily the users learn the interface (learnability).
  - The efficiency of the interface (task performance).
  - How easily the users can memorize.
  - The reduction of errors.
  - The general satisfaction with the interface.

Research so far has shown that the usability factors have an impact on trustworthiness, especially of the websites, as they increase the perceived ability of them (Roy, Dewit, and Aubert, 2001). Moreover, usability is a prerequisite to trust, as users need to trust themselves and their ability that they can use a software system correctly (Sasse, 2005).

- Safety is when there will not be any catastrophic consequences to the users or the environment by the use of the software system (Avizienis, Laprie, and Randell, 2004).
- Maintainability is the ability of the software system to be modified or repaired (Avizienis, Laprie, and Randell, 2004).

All the aforementioned attributes contribute towards system trustworthiness. Some more, some less, depending on the context of the system under development. For example, a user of a system might be more interested in the safety of a software system while another user in the availability. Also, the availability of the system might be more critical for systems controlling financial transactions, while for other systems safety is the major concern. Thus, the context plays an important role in the trust process and as a result it should be considered when trust is analysed and reasoned.

However, the above system trustworthiness definitions are implicitly referring to the characteristics of the software component of an information system, neglecting the human components of an information system or other external software components. Some years ago a system could be trustworthy if for example the software component was secure and usable as the information system basically only depended on that software component. Nowadays information systems are more socio-technical systems and interact with other components, human or system. Thus, the trustworthiness of the system depends also on other human components and external systems.

This is what we believe distinguishes a dependable system from a trustworthy system. In an information system context, dependability refers to the technical competence of the software and hardware components of an information system, such that reliance can justifiably be placed on them. Trustworthiness of an information system however, is not only the dependability of the software and hardware components (Shneiderman, 2000). So, we argue that trustworthiness of an information system is the dependability of the software and hardware components, in terms of security, privacy, etc., plus the consideration of the outcome of the interactions of the software components with other components that are required for the fulfilment



of the information system's goals. For example, even if a software that is part of an information system is dependable in terms of security, the information system will not be trustworthy if one of its users does not keep her username and password safe.

A stakeholder will consider an information system trustworthy if the information system meets his expectations that are considered signs of trustworthiness inside the social setting. Therefore, the stakeholder expectations are translated to functional and non-functional requirements as the above. Therefore, this research refines the definition of system trustworthiness as:

*A trustworthy information system is an information system that fulfils all the functional and non-functional stakeholder requirements. The more trustworthy a system is, the more likely it is that it will fulfil the stakeholder requirements.*

Trustworthiness is the necessary but not sufficient foundation for users to trust systems with justification. It is not enough for the system to be trustworthy, it must also show itself to be trustworthy, i.e. provide evidence of trustworthiness. So, the goal is for the system to be trustworthy so that it will increase the likelihood that users will reciprocate and trust the system.

### **2.3 Trust and Information System trustworthiness in the context of the thesis**

In this thesis the focus is not on the trust relationship between the user of a system and the system. Also, the focus is not on the trust relationships among the components of the system. The focus of this thesis is on the trust relationships between the developer of an information system and the components of an information system. The system-to-be acts as a proxy for the developer once put in operation having the same trust relationships. During requirements modelling the developer is including such components in the requirements model that reflect the behaviour that is expected from them. The further development of the system is based on such expectations and eventually its success in meeting all stakeholder requirements. To this end, in the context of this thesis we define trust as:

*Trust is the positive expectations of the developer about the behaviour of the modelled actors of an information system, from which the technical system-to-be and the whole information system might be positively or negatively affected.*

It is crucial for the success of the system that the developer shows optimum trust to the components of an information system. Trusting less the developer

is losing opportunities to make the system less complex and costly. In contrast, trusting more the developer is setting herself exposed to the information system components' uncertain behaviour. Thus, the developer is becoming vulnerable and eventually the system under development is becoming vulnerable in terms of not meeting its goals. Therefore, the system trustworthiness is dependent upon the satisfaction of the goals of the system assigned to achieve by itself and the outgoing dependencies of the system on the rest of the components of the information system. As a result, in the context of this thesis we define then system trustworthiness as:

*System trustworthiness is a characteristic of the system-to-be that shows to which degree the system-to-be can achieve the goals that have been assigned and to which degree its outgoing dependencies are resolved through trust and control.*

In section 3.4, we describe how we justify the resolution of dependencies through trust and control. In brief, trust and control are the means for the developer to feel confident that the modelled components will behave as expected once the system is put in operation.

## 2.4 Information Systems development methodologies

The British Computer Society Information Systems Analysis and Design Working Group defines a software methodology as "recommended collection of philosophies, phases, procedures, rules, techniques, tools, documentation, management, and training for developers of information systems" (*BCS - The Chartered Institute for IT*). Development of trustworthy software systems is a complex procedure though, where developers do not have adequate knowledge of reasoning about trust and lack the aforementioned collection. However, a methodology copes with the complexity and reduces risks and uncertainties by rendering the development tasks more transparent and visible (Klopper, Gruner, and Kourie, 2007), thus with the creation of a software development methodology for trustworthy software systems developers will be able to model and reason about trust. In general, a software development methodology includes the following steps:

- Requirements elicitation.
- Analysis.
- Design.
- Implementation

- Testing.

This research will focus mainly on the requirements, analysis and design stages, and how the developed designs can be verified as meeting the trustworthiness requirements. Nevertheless, it will also investigate what outputs from the former stages are best suited for the next stages. In addition, the methodology should follow a traditional methodology pattern for software development and not having an ad-hoc approach that will be difficult for developers to adopt it (Presti et al., 2006). Moreover, for a methodology to be characterised as a good methodology (Berard, 1995) requires that it:

- Can be described quantitatively and qualitatively at the same time.
- Can be used repeatedly and achieving the same results every time.
- Can be taught to others easily and in a reasonable time frame.
- Can be applied by others with a reasonable level of success.
- Can be applicable in a relatively large percentage of case studies.
- Can achieve significantly and consistently better results than either other techniques or ad hoc approaches.

In the development of a methodology some characteristics have to be considered. First of all, the methodology should consider all aspects that affect trust and not only a subset (Presti et al., 2006). For example, until now only subsets of trust properties are considered, while neglecting the holistic nature of trust.

Finally, trust should be considered from the early stages of the development process (Yu and Liu, 2001) in order not to create any conflict with security and the other functional requirements of the system (Mouratidis and Giorgini, 2007).

#### **2.4.1 Modelling language**

A system under development can be modelled at different levels of abstraction or from different perspectives. A modelling language contains the elements with which the model can be described. The grammar of the modelling language describes its semantics, syntax, and notation (Karagiannis and Kühn, 2002). A modelling procedure describes the steps applying the modelling language in order to create models, which essentially are the results of the procedure. Mechanisms are algorithms that can be applied to the models. In Figure 2.2 the components of modelling methods are depicted and it is used as a reference model in order to develop the modelling language proposed in this thesis.

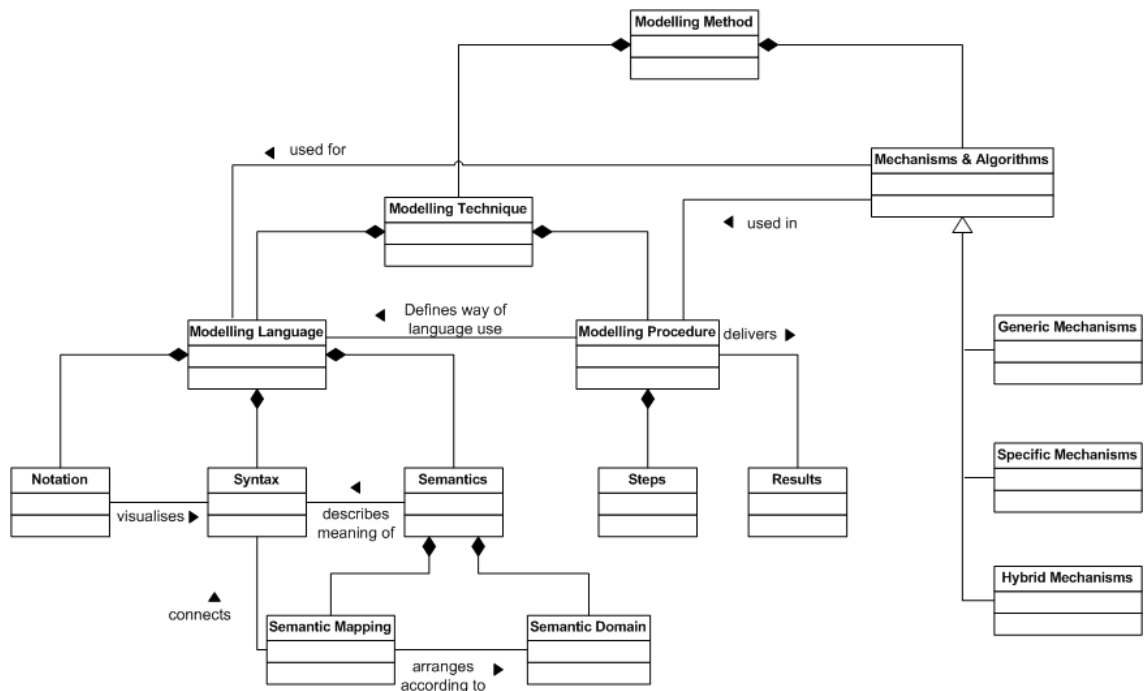


Figure 2.2: Modelling methods components

For modelling languages two major approaches exist to describe their grammar, graph grammars and meta-models. Often, UML class diagrams are used to describe the meta-model of the grammar. For grammatical rules that cannot be fully expressed by class diagrams, additional constraint languages are employed such as OCL. In our thesis, we use UML class diagram to describe the meta-model of our modelling language. The semantics describes the meaning of a modelling language and consists of a semantic domain and the semantic mapping. The semantic domain describes the meaning while the semantic mapping connects the syntactical concepts with their meaning defined in the semantic domain. In this thesis informal textual descriptions are used to define the semantics of the proposed modelling language. The notation describes the visualisation of a modelling language. In this thesis we define symbols for visualising the syntactical concepts of the modelling language. The modelling language mechanisms provide the functionality to use and evaluate the models built using the modelling language. In this thesis mechanisms are proposed to evaluate the trustworthiness level of the technical system under development.

### 2.4.2 CASE tools for Information System methodology

In software engineering there was always the problem of lack of common understanding between the developers of the system and the stakeholders. As a result, graphical modelling has been arisen in order to promote a better communication between the developers and the stakeholders that will eventually establish a common understanding of the domain under inspection. Therefore, graphical models are commonly used to describe and capture functional properties of the environment and the system, as well as to design the system.

However, graphical modelling most of the time is informal and does not contain any logic. Hence, formal frameworks have been developed that combine graphical notations with logic. These tools, called Computer Aided Software Engineering (CASE) tools support the process of software development and they are highly interactive and graphic-intensive (Phillips et al., 1998).

The graphical user interface (GUI) of a CASE tool is usually based around a working canvas. In this area the user produces graphical structures that consist of predefined symbols that usually appear in a toolbox (Phillips et al., 1998). The structures are used to model the system under development and, particularly in the case of this dissertation, the environment in which the system will operate as well.

The elements that can be drawn on the canvas are usually divided into two categories. The entities that represent units of information and the relationships that link the elements. For each CASE tool there are certain well defined rules concerning the appearance of each element. Moreover, the relationships might have attributes on their bodies and on their ends and there is usually a restriction of the way elements can be connected (García-Magariño and Gómez-Sanz, 2008). In detail the functional requirements of a CASE tool are the following (Phillips et al., 1998):

- Insert elements in the working canvas. Once an element is selected from the tool box it should be possible to be inserted in the working canvas and also to enter its properties.
- Edit existing elements. The element or a group of elements should be selectable so as the user to be able to take a variety of actions, such as copy or cut and paste them, edit their properties or delete the elements.
- Create links between the elements. The links will describe the relationship between the elements.
- Edit existing links. The user should have the ability to select links and edit their properties or delete them.

- Edit diagrams. The user should be able to re-arrange the diagrams or part of them, and sometimes place annotations.
- Passive browsing of diagrams. The user should be provided with the ability to move, zoom or rotate diagrams and hide parts of the displayed information according to his will.

On the other hand, the non-functional requirements of CASE tools are:

- Ease of actions. The user should be able to carry all his possible actions easily (Phillips et al., 1998).
- Flexibility of actions. The CASE tool should assist the developers to create quality designs without being too restrictive to a point that will alienate the developers. Therefore, there should be an optimisation of the constraint environment (Brooks and Scott, 2001).
- Prevention of errors and ease of recovery with the inclusion of multilevel "undo" and "redo" actions (Phillips et al., 1998).
- Aesthetically pleasing screens that comply with usability rules (Phillips et al., 1998).
- Effective help system, by speaking the developer's language. The help should be in words that the developer is familiar with (Seffah and Rilling, 2001).
- Quality feedback at each stage or even in some actions, again by speaking the developer's language as mentioned above (Seffah and Rilling, 2001).

CASE tools usually follow the approach of a separated logical model from the views of that model. This approach is based on the fact that an element might appear in two diagrams, but actually is one single element. Otherwise, an element that appears in two different models would be difficult to be processed. Consequently, CASE tools should keep separately the logical model and the views (García-Magariño and Gómez-Sanz, 2008).

The ultimate goal of a CASE tool is to place, join and manipulate the elements of the methodology quickly and easily. Also, the tool should fully support all the aspects of the methodology and must enforce its rules, for example by not allowing the user to apply illegal relationships between the entities (Phillips et al., 1998). Finally, the CASE tool should provide developers the ability to automate many of their actions (Finnigan, Kemp, and Mehandjiska, 2000), so as to reduce the amount of time and money spent on projects and improve the quality of the finished product and its documentation (Finnigan, Kemp, and Mehandjiska, 2000).

## 2.5 Evaluation methods for software engineering methodologies

An essential part of method development is the evaluation of the method, which includes also an investigation of its validity. The validity of a method is more an empirical than a theoretical question, since a lot of methods that sound reasonable in theory; do not work in practice (Moody et al., 2003). To this end we have chosen an empirical approach to evaluate the main contributions of this research. The demand of empirical studies and their contribution to increasing knowledge in the software engineering domain (Runeson and Höst, 2009; Sjoeborg et al., 2005) is continually increasing. In the software engineering domain, it is difficult to select an appropriate empirical method, which is suitable for a specific research context. Generally, the tools for evaluation of a method are experiment, case study, and survey, which include data collection and analysis (Kitchenham, Linkman, and Law, 1997; Zelkowitz and Wallace, 1997). However, evaluations are expensive and there are no mandatory requirements on methods or tools in order researchers to validate their methods or tools (Kitchenham, Linkman, and Law, 1997). Another challenge is the availability of resources such as budget, time, and personnel, in order to relax the idealised assumptions, which are made during the development of a method in order to enhance the insight and the ability to reason (Wieringa and Morali, 2012; Easterbrook et al., 2008), much as possible until the method is tested in a real environment.

There is no one evaluation method that is always the best, but there are many methods each of which is appropriate in different situations (Kitchenham, Linkman, and Law, 1997). The DESMET project (Kitchenham, Linkman, and Law, 1997; Kitchenham, 1996) identified nine evaluation methods:

1. Quantitative experiment. This is an evaluation that includes many subjects who are asked to perform a task using different methods or tools under investigation and is aimed at establishing measurable effects of using a method or tool.
2. Qualitative experiment. This is an evaluation that includes many subjects who are asked to perform a task using different methods or tools under investigation and aimed at establishing that a method or tool is appropriate for specific needs. In this case, the appropriateness of the method or tool is assessed in terms of the required features provided by the method or tool.
3. Quantitative case study. This is an evaluation where the method or tool un-

der investigation is tried out on a real project and is aimed at establishing measurable effects of using a method or tool.

4. Qualitative case study. This is an evaluation where the method or tool under investigation is tried out on a real project aimed at establishing that a method or tool is appropriate for specific needs.
5. Quantitative survey. This is an evaluation where humans that have used a specific method or tool are asked to provide information about the method or tool and is aimed at establishing measurable effects of using a method or tool.
6. Qualitative survey. This is an evaluation where humans that have used a specific method or tool are asked to provide information about the method or tool and is aimed at establishing that a method or tool is appropriate for specific needs.
7. Qualitative effect analysis. This is a subjective assessment of the quantitative effect of a method or a tool based on expert opinion.
8. Benchmarking experiment. This is an evaluation based on a number of standard tests using alternative tools and assessing the relative performance of the methods or tools.

In this thesis, we have chosen to evaluate our proposed methodology using a qualitative case study from the e-health care domain in England in order to observe the effects of our proposed methodology. Additionally, to further validate our work we have chosen to carry out a quantitative and qualitative survey by asking academics, industry researchers, and postgraduate students to use our proposed methodology and support tool and provide us with feedback about the effectiveness of the methodology.

## 2.6 State of the art in trust engineering

The state of the art in the area of trustworthy information system development is very broad and looked from different angles and at different levels. Sutcliffe (2006) argues that information system design and trust intersect in two ways. Firstly, if the design of a system is not thought-out prior to the development stage, then there is a possibility that the system will not be built as per the user's requirements. When the user actually utilises the system, she will be made aware that her requirements have not been fulfilled, hence possibly causing distrust of the system and the possible rejection of it. The second way that design and trust intersect is by having



technology acting as a mediator of trust between people, organisations or between artificial agents who represent humans or organisations. Technology in our days allows us to communicate and collaborate with people around the world. But also, it prevents people from interacting face to face, where they are more able to determine the trustworthiness of the other party. So, there is a need for systems that will enable the collaborating parties to assess the trustworthiness of each other using technology as a mediator (Jøsang, Keser, and Dimitrakos, 2005). Trust is therefore embedded in information systems in multiple ways (Table 2.1). There are lines of research that are focusing on trust modelling during the requirements analysis stage, trust management analysis, and consideration of trust during security requirements analysis. In addition, a branch of human computer interaction research aims to transmit the appropriate trust signals to the users and improve the trust decision process or to increase the trust perceptions. Furthermore, there is a line of research named Trusted Computing, which is focusing on developing trustworthy hardware components that ensure the trustworthy behaviour of other components. Finally, computational trust deals with the development of trust models that can be used by artificial agents to reason about trust.

Table 2.1: Trust engineering approaches

Category	Section	Description
Trust modelling	2.6.1	Provides trust awareness to systems
Trust management	2.6.4	Enables the assessment of trustworthiness
Security engineering	2.6.2	Considers trust during security analysis
Goal satisfaction reasoning	2.6.3	Reasons about the satisfaction of system goals
Human computer interaction	2.6.5	To send appropriate trust signals or manipulate user trust perception
Trusted computing	2.6.6	To enforce trustworthy system behaviour
Computational trust	2.6.7	Trust models used by artificial agents

### 2.6.1 Trust modelling

One line of work on trust adopts the approach of embedding trust in an information system by treating trust as a non-functional requirement, for example such as security or usability. Through this approach developers are guided to capture customer

needs and requirements regarding trust and to reflect them in the functionality of the system. Then it can be argued that users will trust the system once it will be put in operation as it possesses the desired trust requirements from the perspective of the user. Such approaches extend existing software engineering approaches, by enhancing them with notations and processes to represent and reason about trust as a non-functional requirement.

A first approach is by Yu and Liu (2001) that addresses the issues of trust at the requirements level of the system development process. Trustworthiness is modelled as a softgoal, which does not have clear criteria for satisfaction. *i\** concepts, such as actor, goal, softgoal, dependency, are used in order to model the relationships between the components of an information system. The strategic dependency model describes the network of dependencies, while a strategic rationale model describes the reasoning of each actor about her goals. The trustworthiness of the dependee is modelled as a softgoal for the depender and is refined from her viewpoint. So, trustworthiness is modelled as an objective of stakeholders, which influences their dependencies and goals satisfaction. The refinement of the trustworthiness goal represents the rationale for trust and leads to specific requirements for the system-to-be. Therefore, trust is considered as a non-functional requirement, where trust is a combination of all or some quality attributes of the system under development. Furthermore, control mechanisms can be added to relieve the need for trust, which contribute to the viability of the dependency. We adopt the principle from Yu that dependencies introduce vulnerability for the depender, if the dependee does not fulfil the dependency, and that the developer is led to question the viability of the dependencies when considering a network of dependencies as a basis upon which the system-to-be will be developed. Therefore, we add on top of that, that a dependency is also a potential vulnerability for the developer as well.

Secure Tropos (Giorgini et al., 2005) extends Tropos methodology (Bresciani et al., 2004) with the concepts of ownership, trust, delegation, permission, and monitoring. Ownership between an actor and a service exists if the actor is the legitimate owner of the service. Trust between two actors exists when one actor trusts another actor for a certain goal. Delegation between two actors exists when one actor delegates to another actor the execution of a task or access to a resource. Moreover, the authors define two types of delegation, delegation of permission and delegation of execution. In the first type of delegation the dependee is authorised to achieve a goal but she does not have to, while in the second case the dependee has to achieve the goal. The concept of trust is introduced in order to capture the existence or non-existence of trust in the cases of delegation, since sometimes actors delegate goals to actors that they do not trust as long as there are ways to hold

such dependees accountable. Similarly to delegation, there is trust of permission and trust of execution. In the first case the depender trusts that the dependee will not go beyond the achievement of the goal, while in the second case the depender trusts that the dependee will at least achieve the goal for her. Furthermore, in cases where there is no trust, the concept of monitoring is introduced. The act of monitoring can be done by the delegator himself or by another actor who plays the role of monitor in order to check for the violation of trust. For example, in Figure 2.3 Bob owns his personal information and while providing that information to Sam there is a trust of permission that Sam will not misuse his private information. Alice wants from Sam personal information for statistical reason and there is trust of execution that Sam will at least fulfil this goal. If Alice does not trust Sam to fulfil the goal then she can delegate the goal to monitor Sam to Carol. Carol then will monitor if Sam is fulfilling the goal to provide personal information to Alice. Through this way, the developer can capture trust relationships in a normal functional requirements model, and with the introduction of the mentioned concepts security and trust requirements are deriving. We agree with the authors that sometimes there may be dependencies on other components that are not trusted, as long as there are ways to hold those dependencies accountable. To this end, monitor is a solution to check if dependencies are being fulfilled.

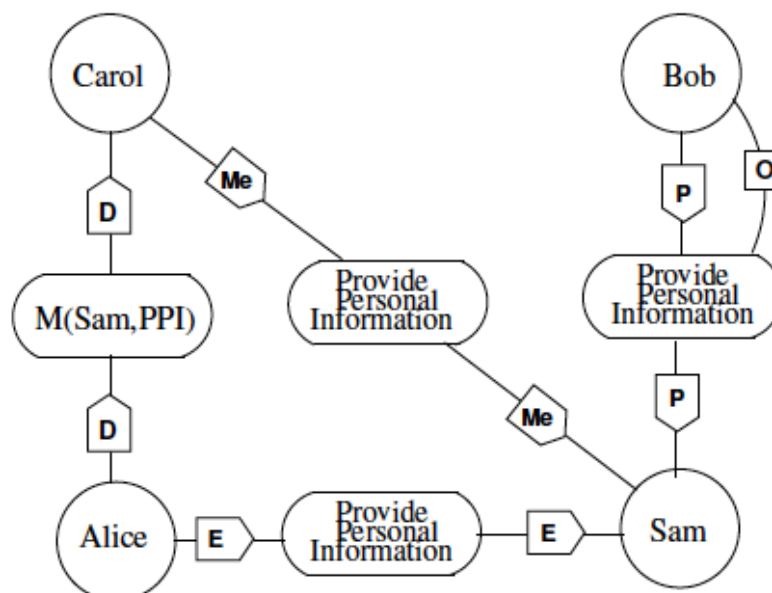


Figure 2.3: Monitoring example

Bimrah (2009) extends the Secure Tropos (Mouratidis and Giorgini, 2007) methodology with the concepts of request, action, trust relationship, trusting intention,

reputative knowledge, recommendation and consequence in order to model trust (Figure 2.4). Request is the act when an actor is asking for something to be done or given by another actor, while the activity of the second actor, as a response to the request of the first actor, is the action. A trust relationship indicates that one actor expects another actor to behave in a certain way. Trusting intention is defined as the intent of the trustor on how far she actually trusts the trustee to carry out the action to her request. Reputative knowledge is the knowledge that the trustor has about the trustee. Recommendation is the representation in favour of another actor while consequence is the effect of a trust relationship. The developer is guided through a series of models, using the aforementioned concepts, in order to analyse and reason about trust relationships. The first model is the Request/Action Model (R/AM), which models the action and the request, along with the resources and security constraints that are put upon the actions and request. The next model is the consequence model, which models the possible consequences of a request, which can be positive or negative. The model that follows is the recommendation model, which shows the recommendation for an actor. The consequences model, the trusting intention and the reputative knowledge of the trustor towards the trustee influence the recommendation of the trustee. The final model is the Trust Relationship Model, which shows whether an actor trusts another actor, what is the trust level, and whether other actors should trust the trustee, which ultimately help the developer make design decisions. The author's proposal captures in detail why one actor trusts another actor.

In (Yan and Cofta, 2003) the system analysis and design considers different domains in mobile communications. A trusted domain is a set of domain elements such that all domain elements share certain defining statements regarding their trust definition, which must be fulfilled in order an element to be trusted. Even though the trust definition is common among the elements of a domain, however, there are trust gaps between trusted domains because of the subjectivity of trust definitions. To address this, certain elements bridge the gap and are responsible for ensuring trust at a higher level than the one of the domains. The component that is trusted by more than one domain and is acting as a bridge between them is named trusted bridge. The methodology contains four steps: model the mobile communication system and identify the different trusted domains and their entities, analyse each domain to identify the trust statements, identify bridging solution for domains that do not share any trusted component, and form the trusted bridge. To form the trusted bridge the developer can use an existing component, create a new one, or create a new separate domain that will bridge the two original domains as shown in Figure 2.5. The methodology can be applied to any system analysis and we

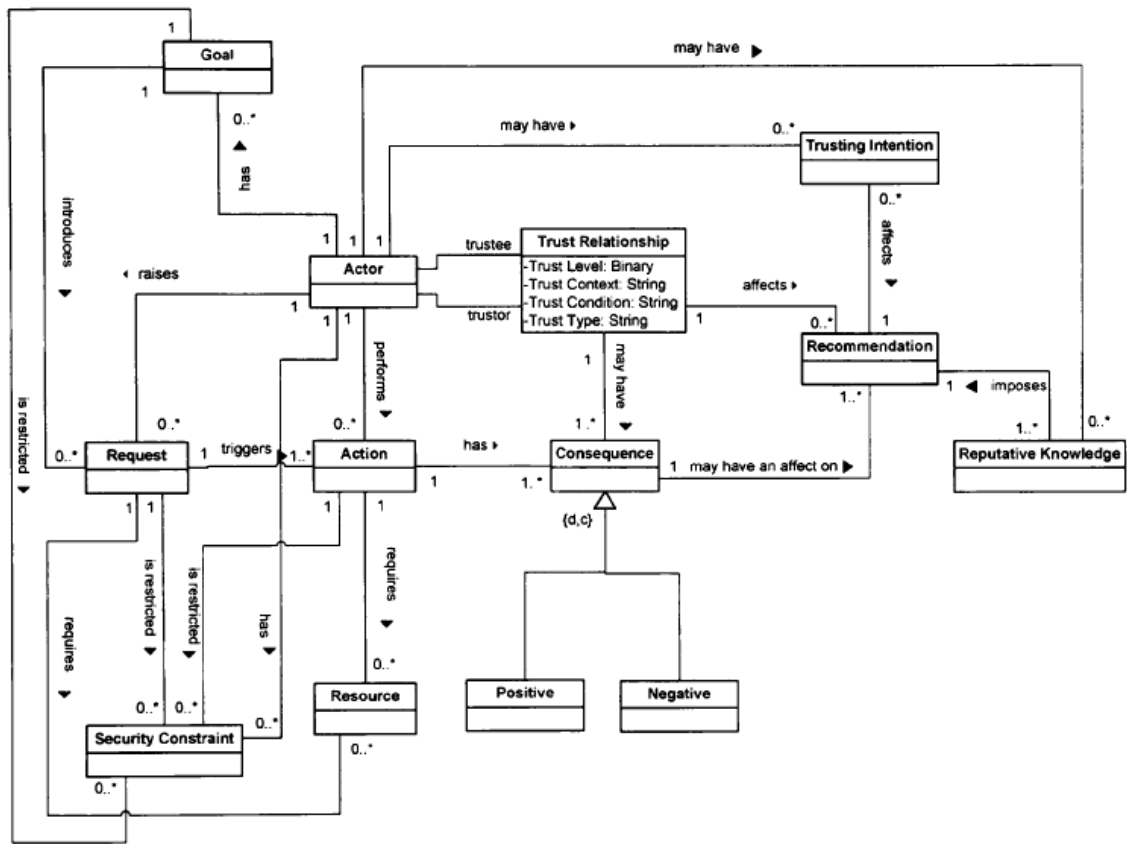


Figure 2.4: Bimrah meta-modell

agree here with the authors that it is vital for the system analysis not to contain any gap in the chain of trust relationships especially between the developer and the components of the system.

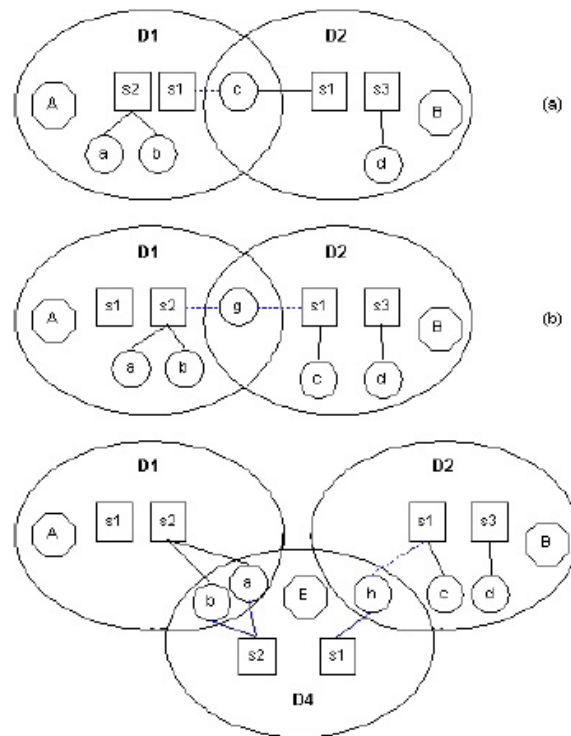


Figure 2.5: Methods for bridging trusted domains

Presti et al. (2006) describe a holistic methodology to analyse trust during the development of the system that focuses on the user of the system. The methodology contains five steps as shown in Figure 2.6. The first step is the identification of scenarios, which are short narratives that describe the user's behaviour focusing on the use of services provided by the system under development. Individuals external to the trust analysis and the system design then validate the scenarios. The second step is the trust analysis using the Trust Analysis Grid. The rows of the grid represent aspects of the system described by one or several sentences of the scenarios, while the columns represent categories of trust issues, subjective; system; and data; that the authors have derived by studying the of the art in trust. The developer then checks the ones that are satisfied in those aspects of the system. The completion of the cells is done with a number of X or Y marks, the name of a more precise issue, or a signed number, that indicate the importance of the trust issue to the specific aspect of the system, the more precise issue, or the scale of trust issue respectively. The third step involves examination of the previous trust issue by peers of the developer

and from the perspective of another potential user. The fourth step contains the refinement of the scenarios based on the previous peer review, and the last step is the construction of guidelines for identifying trust requirements based on the Trust Analysis Grid constructed previously. The guidelines are derived from the insight that was achieved during the previous four steps.

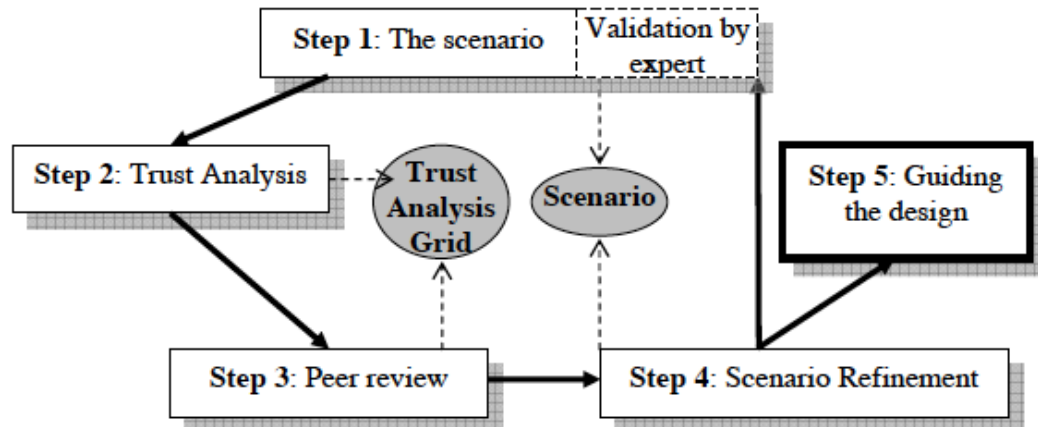


Figure 2.6: Holistic trust analysis process

This direction of research investigates trust from the perspective of the users or other stakeholders attempting to capture their trust related needs and requirements. However, trust requirements are not a subset of requirements but all possible requirements. There cannot be a requirement that it is not a trust requirement. Furthermore, we investigate the trust relationships, which come into existence because of the dependencies, not from the perspective of the depender but from the perspective of the developer. These are equally important for the developer, as the system-to-be will be built according to that configuration of network of dependencies. If the developers models dependencies and trust relationships that are not valid then the system will not be built on solid foundations and there might be potential vulnerabilities, which will constitute the system untrustworthy. Regarding this aspect, the approaches mentioned in this section fail to provide support to the developer. Moreover, there is limited support in providing required abstractions that will enable the developer to enforce the fulfilment of dependencies in cases where there is not trust. Furthermore, the previous approaches they do not look into whether the system will be able to do what it is supposed to do in order to be trustworthy because they do not reason about the system interactions that are necessary for the satisfaction of the system goals. Therefore, trust in the system is partially blind, as it has not been fully justified. So, the focus is on trust modelling

but without supporting the developer in reasoning about trust relationships inside an information system that are important to the system trustworthiness. The developer is left without guidance and as a result trust relationships from the perspective of the developer are not investigated.

More particularly in (Yu and Liu, 2001), there are no constructive techniques that can guide the developer in the refinement of the trustworthiness softgoal, for example what are the aspects of trustworthiness. In addition, there are no control related concepts and techniques available to guide the developer in identifying and analysing the appropriate control mechanisms for the system-to-be. Giorgini et al. (2005) fail to provide method for the developers to reason why there is trust in a specific relationship. Also, even if monitoring mechanisms are in place still they are not enough to enforce the fulfilment of a dependency. Furthermore, indirect trust relationships are omitted from investigation, for instance, in case of monitoring dependency, which is actually an indirect trust relationship, it is left unexamined and there is not analysis on whether there is trust. If it proves that there is no trust then the system will not achieve its designed functionality. Therefore, indirect trust relationships stay hidden without proper justification and assessment and becoming a serious threat for the proper operation of the developed information system. In (Bimrah, 2009), although it provides a mechanism for reasoning about trust relationships, it fails to capture indirect relationships as well. For example, should an actor trust the recommendation provided from other actors. In addition, there are no constructive techniques to guide the design of the system in case of lack of trust in a dependency. In (Yan and Cofta, 2003) there are no constructive techniques available on how the developer will identify the components that will act as a trusted bridge or how she will create one or create another domain that will bridge the two original components. Finally, Presti et al. (2006) identify what the system must do in order the user to trust it. However, a trust requirement is not a subset of the system requirements, but it is the system requirements and even more. Moreover, even though the approach forces the developer to identify trust issues, this approach does not provide concepts and techniques in terms of supporting the developers in reasoning about trust.

### **2.6.2 Security engineering considering trust**

This line of research includes contributions that even though some of them do not claim that their goal is to achieve system trustworthiness, nevertheless, they significantly improve system trustworthiness in terms of security, which is considered widely as one very important aspect of trustworthiness. Also, in this category fall



approaches that deal with domain knowledge and trust in order to achieve security. Trust is an enabler of security because all security services rely to a great extent on some notion of trust (Viega, Kohno, and Potter, 2001; Ray and Chakraborty, 2004; Haley et al., 2006). Secure systems have been built under the premise that concepts like trustworthiness or trusted are well understood, unfortunately without even agreeing on what trust means. Therefore, such works have investigated the concept of trust in order to build secure systems.

In (Górski et al., 2005) a trust case represents a complete and explicit argument that influences trust in the system under development in terms of security and safety. The trust case is decomposed into claims, using the Claim Definition Language (CDL), that posit trust related properties and then follows the collection and production of supporting evidence and development of a structured argument that the evidence is supporting the claims. The evidence can be a fact, which is a statement of verified information about something, an assumption, which is a statement assumed to be true for which there is no supporting material, or another claim for which the same procedure needs to be followed. The trust cases can be modelled using UML stereotypes that influence the trust level of the trustor in the system. Also, a UML based graphical language is used to represent the context in which a specific claim is interpreted. Figure 2.7 depicts the conceptual model of a trust case. We agree with the authors that security and safety are two important properties of system trustworthiness, and especially that it of paramount importance that trust in the system under development needs to be justified by considering the trust assumptions that underlie the system development.

Haley et al. (2006); Haley et al. (2008) investigate trust assumptions in the context of analysis of security requirements. This framework consists of four activities: identification of functional requirements; identification of security goals; identification of security requirements; and construction of satisfaction arguments; In the last activity the developer constructs two satisfaction arguments, named outer and inner arguments, that enable her to identify incorrect assumptions about security related system components. The outer arguments are claims for the system environment expressed in formal logic, in particular that the system environment is correctly defined and that the system will not introduce any undesired behaviour by the system components. The inner arguments are informal arguments that support the claims in the outer arguments and are based on trust assumptions. Trust assumption is defined as "a statement about the behaviour or properties of the world the system lives within, made in order to satisfy a security requirement and assumed to be true". The process includes the annotation of the system environment with the relevant phenomena between the system component and the specification of the de-

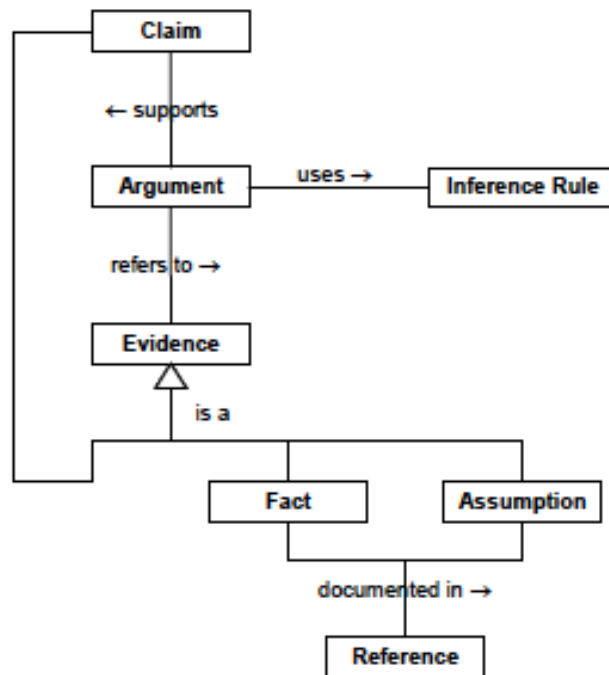


Figure 2.7: Trust case conceptual model

sired behaviour from the components along with their justification. We agree with the authors and we recognise as well that to say there will not be any undesired behaviour by a system component is a very strong assumption and that we cannot totally ensure the trust of the assumptions, still though it enables the system to be built on strong foundations in order to be as trustworthy as possible not only in terms of security though but in terms of its full functionality.

Elahi and Yu (2009) propose a method for discovering trade-offs that trust relationships bring between trust and control. The method contains seven steps: identification of actors and their dependencies; modelling and reasoning of actors' goals; modelling trust relationships; recording trust rationale; replacement of the trustee party with a malicious party; analysis of vulnerabilities; and analysis of the trade offs. In this approach the trust modelling techniques from  $i^*$  (Yu, 1995) and (Giorgini et al., 2005) are adopted and the trust rationale is captured as a belief from the viewpoint of the depender in order to reveal implicit trust assumptions. Then the dependee entity is replaced with a potential malicious entity that has the same access and capabilities as the legitimate entity. As a result the developer can then model and analyse the vulnerabilities and their impact that the potential malicious entity may bring. The analysis is a cost/benefits analysis where the cost is the risks that the malicious entity is bringing. The aim of the trade off analysis is to

evaluate if the potential vulnerabilities because of lack of trust in the entities that have been assigned with goals outweigh the benefits of the dependency relationship. If the potential vulnerabilities outweigh the benefits then the developer can choose an alternative dependee that offers better ratio of benefits and vulnerabilities. The costs and benefits of each alternative dependee are evaluated in terms of satisfaction or denial of top goals of the depender.

The following security engineering methods aim towards the development of secure systems. Security is a main aspect of trustworthiness and has a unique role in the establishment of trust in the information systems. For example, a user might use and trust a system even though it does not have certain functionality but it is most probable that he will use and trust a system that it is not secure.

Trustworthy Computing (Lipner, 2004; Charney, 2012) is an initiative by Microsoft, which uses the Security Development Lifecycle (SDL) for the development of their software that must withstand attacks. The initiative is concentrated on planning activities in the areas of security, privacy, reliability, and business integrity. Security objectives should be identified along with security feature requirements that are based on customer demand and compliance with standards. Then additional security features are identified as part of threat modelling. More specifically, threat modelling includes four steps: identification of use scenarios; identification of assets; identification of threats; and identification of countermeasures. Threat modelling is carried out component by component followed by the identification of the assets the system-to-be will manage along with the interface used to access them. Then the threats for these assets are identified followed by the countermeasures that mitigate the risk and protect the assets. The countermeasures can be in the form of security features, such as encryption or access control, or in the form of proper function of the system.

Secure Tropos (Mouratidis and Giorgini, 2007) is an extension of Tropos methodology (Bresciani et al., 2004) and is based on the concept of security constraint to analyse the security requirements. With Secure Tropos, the security requirements can be modelled, reasoned, and transformed into a design that satisfies them. New concepts, such as security constraint, secure dependency, and secure entity, are introduced in order to enable the developer to analyse the security of the system under development. Security requirements are captured as security constraints that represent a security related restriction in terms of confidentiality, integrity, and availability. Secure dependency is a dependency with security constraints that restrict its fulfilment unless the security constraints have been satisfied by the depender or the dependee. Secure entity is a secure goal, plan, or resource. First, the developer models the system-to-be and the actors of system environment and then she models

the security constraints imposed to the actors. Then, secure entities are modelled by identifying the secure goals, plans, and resources that are satisfying the security constraints. Secure Tropos was extended in (Pavlidis, Mouratidis, and Islam, 2012) in order to consider domain knowledge and reason about trust relationships with components of the environment of the system. In particular, trust and control resolution of dependencies on actors with secure goals are identified in order the developer to build confidence that such a component of an information system will fulfil its secure goals once the system is put in operation to ensure its security.

KAOS (Van Lamsweerde et al., 2007) is a goal-oriented requirements engineering methodology that supports developers in understanding what are the requirements of the system under development. Its conceptual model, the associated language and its techniques enable the developer to identify functional and non-functional requirements, including security requirements. The elaboration of security requirements is performed using anti-models, where anti-goals are used to capture the behaviour of the potential attackers and they represent a threat to the security goals of the system. The anti-models are constructed once the goals of the system under development have been defined. Then the developer derives new security goals as countermeasures to counter the anti-goals of potential attackers. The security goals assigned to components of the system-to-be environment are expectations, which are essentially assumptions.

In (Hatebur, Heisel, and Schmidt, 2007), authors defined patterns for structuring, characterising, and analysing problems that occur frequently in security engineering, which are named security problem frames and they are served to analyse security related requirements. Security problem frames are special types of problem frames (Jackson, 2001), which refer to the problems concerning security and consider security requirements. The transformation of security requirements into concretised security requirements is achieved by selecting security mechanisms, which is essentially the security solution. Every security problem frame is constructed based on a security problem frame template that consists of the following fields: name, which specifies the kind of security problem frame; frame diagram, which shows the domains, their interfaces, and the security requirements; security requirement, which states the security requirement informally; declarations, which are necessary entities for stating the preconditions and postconditions; preconditions, which need to be met by the environment in order the frame to be applicable; postconditions, which are formal representation of the security requirement; and related, which are related security problem frames; The preconditions are essentially the assumptions that need to be true in order the security requirements to be met. In order the developer to guarantee that the preconditions hold there are two alternatives. Either

to be assumed true or they have to be established, or in other words enforced, by using another security problem frame whose postconditions match the preconditions of the initial frame.

CORAS (Lund, Solhaug, and Stølen, 2010) is a method for model driven security risk analysis of a system. It consists of a specialised language for communication, documentation, and analysis of security threats and risk scenarios. In the beginning it was defined as a UML profile but it was later refined and customised. The CORAS method includes seven steps (Figure 2.8). In the first step, which is an introductory step the target, scope, and the size of the analysis are specified in order the necessary preparations to be made. The second step includes the specification of the targets that need to be protected and a common terminology to be used by the developers and the customers. The third step establishes a more correct and refined understanding of the target and the objectives of the customer in order to eliminate any misunderstandings. Moreover, assets are identified and a high-level risk analysis is carried out. Step four includes the agreement between the developer and the customer on the target to be analysed, including the scope, focus, and all assumptions. The fifth step is the risk identification, followed by the sixth step the estimation of risk level of risks. The seventh step is the risk evaluation, while the last step is the risk treatment identification and analysis. In step four is when assumption are being made, and the authors define assumptions of a risk analysis what we take for granted or accept as true, although it may actually not be so. They also state that an assumption is something for which there is strong evidence or high confidence in. The language used for documenting and reasoning about assumption is named dependent CORAS since it is used to document dependencies on assumptions being made. Assumptions in CORAS are used as a means to choose the desired or appropriate focus and scope of the analysis, by assigning a likelihood level to the likelihood of an assumption to occur.

Moreover, UMLsec (Jürjens, 2005) is an extension of Unified Modelling Language (UML). It is a UML profile that enables the developer to express security properties on design models. Standard UML extension mechanisms in the form of labels are used such as stereotypes together with tags to formulate the security requirements and assumptions on the system-to-be environment. More specifically stereotypes and tags represent a set of desired properties. The developer can use the labels to give a specific meaning, with respect to security, to elements of a design model.

The approaches presented in this section mostly focus on the security of the system and in that context consider assumptions and trust. However, despite the fact that most of the approaches acknowledge the importance of trust in the achievement of security, they do not offer a trust assumption reasoning method for the

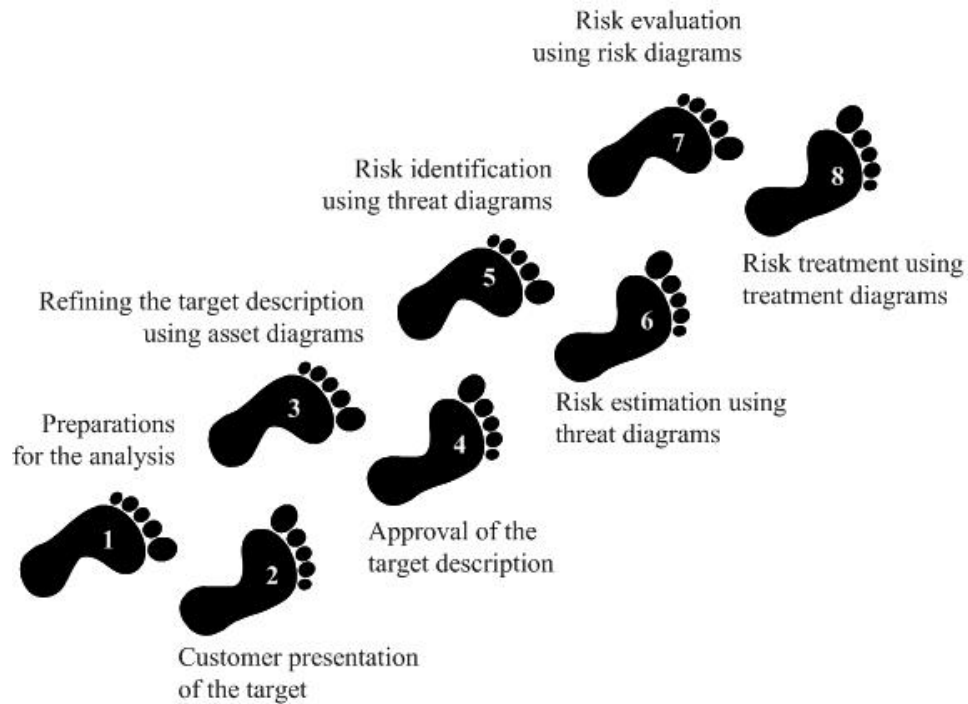


Figure 2.8: CORAS' steps

developers.

More particularly, (Górski et al., 2005) trust cases were only focusing on the security and safety of the system, excluding important other trust properties such as privacy and usability. Even though the development of the context model enables the identification of the components that are involved in trust assumptions, there is no systematic and structured method for identifying specific assumptions. Thus, the developer has to identify the specific trust assumptions in an ad-hoc way, possibly omitting important ones. Furthermore, this approach was limited in identifying only the assumptions for components with whom there are direct trust relationships, omitting assumptions about indirect trust relationships. Also, there were no constructive techniques for alternative solutions in case the assumptions are not valid. Haley et al. (2008) approach has been developed for the purpose of ensuring the satisfaction of only security requirements and not for the rest of the requirements. Furthermore, this approach does not provide a systematic process for annotating the context with phenomena from which the desired behaviour of components is derived. Thus, some trust assumptions might be omitted by the developer. Moreover, there is no support in case of lack of trust, such as requirement abstractions and techniques that will enable the developer to identify system functionality in case of rebuttal that will ensure the desired behaviour of an untrusted

component. Elahi and Yu (2009) approach although it deals with dependencies on other entities and the corresponding trust relationships, it is inadequate in its trust reasoning techniques. It offers modelling concepts to capture trust rationale, but it does not offer a systematic process that can guide the developer in identifying that trust rationale. Furthermore, the developer might not always find an alternative dependee with an adequate ratio of benefits and vulnerabilities. Trustworthy Computing . For Secure Tropos there was no support for the developer to reason about why a component of the information system will fulfil a secure goal. Even though the extension addressed this limitation, it offered limited constructive guidance and alternative solutions. KAOS enables the partial identification of assumptions and even though it provides support for the developer to reason about the ability of a component to achieve a goal it fails to provide support for reasoning about whether the component will achieve a goal. Hatebur, Heisel, and Schmidt (2007) approach lead to partial identification of assumptions as the process does not ensure that the developer will identify all assumptions. Moreover, there is no support in reasoning about such assumptions. However, the assumptions in CORAS are more assumptions about the condition of the physical environment of the system under development and moreover the focus is to use the assumptions in order to choose the desired or appropriate focus and scope of the analysis rather than reasoning about the assumptions. UMLsec (Jürjens, 2005) supports the developer in documenting assumptions on design models. However, there is no systematic process for the developer to identify the assumptions and most importantly the analysis of whether the assumptions hold should be carried out at the requirements stage where it is more cost effective. Finally, the approach related with the extensions of use cases and UML does not support the modelling and analysis of security requirements at a social level, but they treat security as technical solutions (Mouratidis and Giorgini, 2007). Security though is a multidimensional issue that has social characteristics, since the software system will operate in a human social environment and the human factor plays a very important role in security (Mouratidis and Giorgini, 2007).

### **2.6.3 Goal satisfaction reasoning**

At the heart of requirements engineering is the investigation of alternative options and the impact of such options on the system, which generally have different contribution to the degree of satisfaction of top-level goals. To this end various qualitative and quantitative frameworks have been proposed to assist in the assessment of alternatives for decision-making. Also, there has been work on exceptional behaviour of entities upon which goals have been assigned and obstacles are used to repre-

sent such exceptional behaviour. Finally, the state of the art includes approaches that enable the developer to reason about the realisability of a goal that has been assigned to a component.

The NFR framework (Chung et al., 2000; Chung and Prado Leite, 2009) concentrates on modelling and analysis of non-functional requirements. The authors emphasise the need that non-functional requirements need to be identified from the early stages of information system development in order to assist the developer in making design decisions, and also that non-functional requirements should be considered along with the functional requirement throughout the development process. Non-functional requirements are represented as softgoals, which are goals that don't have clear-cut criteria of satisfaction. The non-functional goals are satisfied through the collaboration of the software system behaviour and environment phenomena caused by the rest of the components of the information system. It supports the developers in specifying positive or negative influences of different alternatives on non-functional goals. Softgoals interdependencies are captured with positive ("+" ) or negative ("-") contributions. NFR is used to evaluate and compare the contribution of alternatives to the softgoals qualitatively. By analysing these alternatives, the developer can select the one that best meets top-level quality requirements of the system under development.

A quantitative approach to goal satisfaction reasoning is given by Giorgini et al. (2003), where the authors offer a precise semantics for the relationships between goals which comes in qualitative and numerical form. that is based on a probabilistic model. For every goal there can be full evidence that the goal is satisfied or denied or there can be partial evidence that the goal is satisfied or denied. To this end, two variables are introduced for each goal, which represent the current evidence of satisfiability and deniability of that goal and two constants are introduced that represent full evidence and different level of partial evidence. A probabilistic model has been adopted where the probability that a goal is satisfied or denied represents the evidence of satisfiability or deniability respectively. The starting point of the process is the externally provided assertions as initial conditions. Then the satisfiability and deniability evidence and numeric values are propagated through the goal model according to propagation rules in order to deduct satisfiability and deniability values for the top-level goals. To support the developer the authors propose a label and numeric value propagation algorithm. The authors acknowledge the issue of considering the reliability and competence of the initial source of evidence as the whole process is based on the assumption that the initial evidence is valid.

In (Letier and Lamsweerde, 2004) the authors propose techniques for specifying partial degrees of goal satisfaction and for quantifying the impact of alternative



decisions on the degree of satisfaction on goals. The partial degree of satisfaction of a goal is defined by annotating the goal with quality attributes and objective functions, which are goal related variables and functions that define quantities to be maximised or minimised. Then the authors provide propagation rules that enable the developer to estimate the degree of satisfaction of a higher-goal based on the degrees of satisfaction of its subgoals. To this end the authors provide a catalogue of quantitative goal refinement patterns to assist the developer in specifying goal refinements equations. A bottom-up propagation of quality variables of the low level goals is used to evaluate the alternative system designs, while a top-bottom propagation of quantitative requirements of high level goals is used to specify concrete quantitative requirements of low level goals. The estimations on goal satisfaction assigned to the system under development are quantitative requirements while estimations of goal satisfaction assigned to entities of the environment are quantitative assumptions.

KAOS (Van Lamsweerde and Letier, 2000) has embraced obstacles and provides well-developed methods for detecting and mitigating the obstacles. An obstacle to a goal is defined as an assertion that is consistent with the domain information, but the negation of the goal is the logical consequence of the combination of the assertion and the domain information. In other words, obstacles represent the potential ways in which a system might fail to meet its requirements. Once identified, obstacles are refined similarly as goals in order to identify subobstacles. The goal of the obstacle analysis is to anticipate exceptional behaviour of entities upon which goals have been assigned and to derive more complete requirements and realistic requirements, by specifying alternative ways of resolving such problems early during the development of the system to be. Examples of techniques for obstacle resolution are obstacle prevention, goal substitution, agent substitution, and obstacle tolerance. The process stops when the developer has identified goals that prevent an obstacle or substitute goal and there are no more obstacles or the remaining obstacles are acceptable without resolution.

In (Letier and Lamsweerde, 2002) the authors propose a method to refine goals until they are assignable to single entities and to assign a goal to entity only if the entity can realise a goal. Their proposal also includes a complete taxonomy of realisability problems. In particular a goal is not realisable if there is: lack of monitorability; lack of controllability; reference to future; external unachievability; unbounded achievement; In case there is unrealisability then the developer is guided to address the problems with the help of a catalogue agent based refinement tactics that refine the unrealisable goals and make the realisable.

To mitigate the vulnerability that a dependency is introducing the viability of

the dependency is analysed in (Yu, 1995) by identifying patterns of dependencies that may serve to enforce commitment, assure success, or insure against failure. There if there is a way for the depender to make a goal of the dependee to fail then the commitment is enforceable.

The NFR framework (Chung et al., 2000; Chung and Prado Leite, 2009) does not provide any reasoning technique in the case where a goal that contributes of a top-level softgoal cannot be accomplished by the system and the system is depending on another component of the information system to fulfil it.

In (Giorgini et al., 2003) there not techniques to reason about the trustworthiness of the source of evidence, which means that the analysis is based on unjustified assumptions. Furthermore, even though this approach can be applied to goals that are dependent upon other entities for fulfilment, the issue of trust on the other entity for goal fulfilment is not considered in the goal reasoning process. Reasoning that an entity can achieve a goal does not imply that it will actually show the desired behaviour and achieve it.

In (Letier and Lamsweerde, 2004) the specification of partial degrees of goal satisfaction is based on real data to determine the level of satisfiability, such as statistical analysis of the current system or reliability figures about standard devices, while often it is the case that the developer is required to make decisions that include a lot of uncertainty about the developed system.

In (Van Lamsweerde and Letier, 2000), the developer still faces the problem of reasoning about his trust relationship with the entity upon which a goal, that prevents an obstacle or substitutes a goal, has been assigned.

The main drawback with this approach is that addresses the issue of whether an entity that has been assigned with a goal can realise it, but the problem of whether that entity can be trusted to realise the goal, even thought it can realise it, is not addressed.

However, this only applies in the cases of reciprocal dependencies. If there is evidence that the dependee will fulfil the dependency then there is assurance. For example, there is evidence that the depender and the dependee have common interests related to the dependum. There is an insurance against the non-fulfilment of a dependency if there are alternatives dependees that can fulfil the dependency.

#### **2.6.4 Trust management**

The PolicyMaker approach (Blaze, Feigenbaum, and Lacy, 1996) is proposed for trust management of Internet applications and builds trust relationships between entities. Every entity has a public key and is bind to some credential, where a cre-

dential allows an entity to a specific system environment. Essentially, PolicyMaker is a query engine, which evaluates whether a proposed action is consistent with the trust policy. Also, a next version of Policy Maker is KeyNote (Blaze, Feigenbaum, and Keromytis, 1999), is enhanced in the verification of such polices.

REFEREE (Chu et al., 1997) is a trust management system for web applications, which specifies a language for defining trust polices and provides a general policy evaluation mechanism for web clients and servers. It evaluates requests and returns a tri-value and a statement list, which is the justification of the answer. A tri-value is true, false or unknown. There are two phases, where the first phase, which is called bootstrap, the host application gives the unconditionally trusted assertions and a module database. In the second phase, called the query phase, the host application provides the actions and other arguments such as credentials. Then the interpreter with the policy and the list of arguments is run and then it returns an answer to the host application.

TPL (Herzberg et al., 2000) is similar to PolicyMaker, but permits negative rules preventing access. Every entity may get a certificate from a trusted third party. Then the decisions will be taken based on the evaluation of those certificates that the entities are holding. The trust establishment module validates the client's certificate and then maps the certificate owner to a role. Then the information is sent to another module, which stipulates the access rights that are bound to the particular role.

The Simple Universal Logic oriented Trust Analysis notation (SULTAN) is a proposed trust management solution that allows the developer to perform management of trust relationships. Trust management is defined as "the activity of collecting, encoding, analysing and presenting evidence relating to competence, honesty, security or dependability with the purpose of making assessments and decisions regarding trust relationships for Internet applications" (Grandison, 2003).

Trust management in the previous line of work is mostly focused on access control for resources and especially with authentication and authorisation capabilities. Therefore, the goal of the developer is on defining the trust rules and the ways access control will be implemented and consequently the focus is at a very low level. However, the aforementioned approaches lack guidance on how to write the predicates in policy and certificate assertions that reflect the trust policy and they are designed for servers that are managed by an administrator.

Tan (2003) proposes a matrix model, which can enable trading partners in electronic commerce to analyse trust building services. An e-service is represented in the form of a grid and the grid rows represent a theoretical decomposition of the notion of trust into four reasons namely social signs, personal experience, understanding

and communality. Each reason is split into two sources, according to whether they correspond to trust created by the other party of the transaction, named party trust, or by a control mechanism, named control trust. This analysis is carried out at three different layers: communication with the other party; trade documents and regulation of the transaction; business relationship, which is considered the most important as it can compensate for problem in the previous layers.

TrustCoM is a framework (Wilson et al., 2007) for trust security and contract management for dynamic virtual organisations. It contains a set of semantically well-founded concepts and relationships for describing and reasoning about trust and security in dynamic virtual organisations. In addition it contains an abstract architecture reflecting the previous concepts and providing a flexible structure and organising principles for systems based on the framework. And the third component is profiles extending existing open specifications of services and protocols to implement the TrustCom framework. The framework models trust relationships based on electronic contracts, policies, and reputation systems. Thus it will allow companies to integrate services, processes and resources and form virtual organisations.

Uddin and Zulkernine (2008) present the UMLtrust framework, which considers trust from the early stages of the development process and it is a scenario based analysis of trust. It uses UML, which is well accepted among software developers, and extends it with specialised notation in the domain of trust. A framework is also described that enables the developers to specify the trust scenarios and derive trust rules based on such scenarios. Since the framework is incorporated in the software development lifecycle the developers are able to identify the relevant system requirements for monitoring and trust decision making during the run-time. For example, a server would be able to decide whether a user that requests access should be authorised based on the derived trust rules. However, the framework is inadequate in providing guidance in identifying the trust cases and ultimately the trust relationships.

In (Pourshahid and Tran, 2007) the proposed method makes use of the Goal Requirement Language (GRL) and Use Case Map (UCM) which both of them belong to the User Requirement Notation (URN). Specifically, trust is captured as a soft goal of the trustor because of its uncertainty of whether is satisfied or not and because of its fuzzy nature. First, the UCM is used in order the developer to visually define the trust making process of the trustor and the trust and distrust paths that she can take. Then, the GRL is used in order to model that contribute to the establishment of trust, along with the their threshold values. The threshold values are the points where the trustor will decide between trust and distrust. After the soft goals of the trustor have been investigated, then it is the turn to investigate

how the trustee can increase trust in her. A trust model is provided as a reference (Figure 2.9), which includes three layers. The first layer shows the decomposition of trust into a cognitive and affective type of trust. The second layer shows the further refinement of cognitive and affective types of trust, while the third layer shows some indicative tasks that the trustee can carry out that contribute to the establishment of trust. Further analysis of trust as a soft goal eventually leads to well-defined tasks for the trustee in order to gain the trust of the trustor. The two GRL models of the trustor and the trustee can be combined by the developer in order to show the importance of the trustee's attributes to the trustor and identify the requirements for the system-to-be.

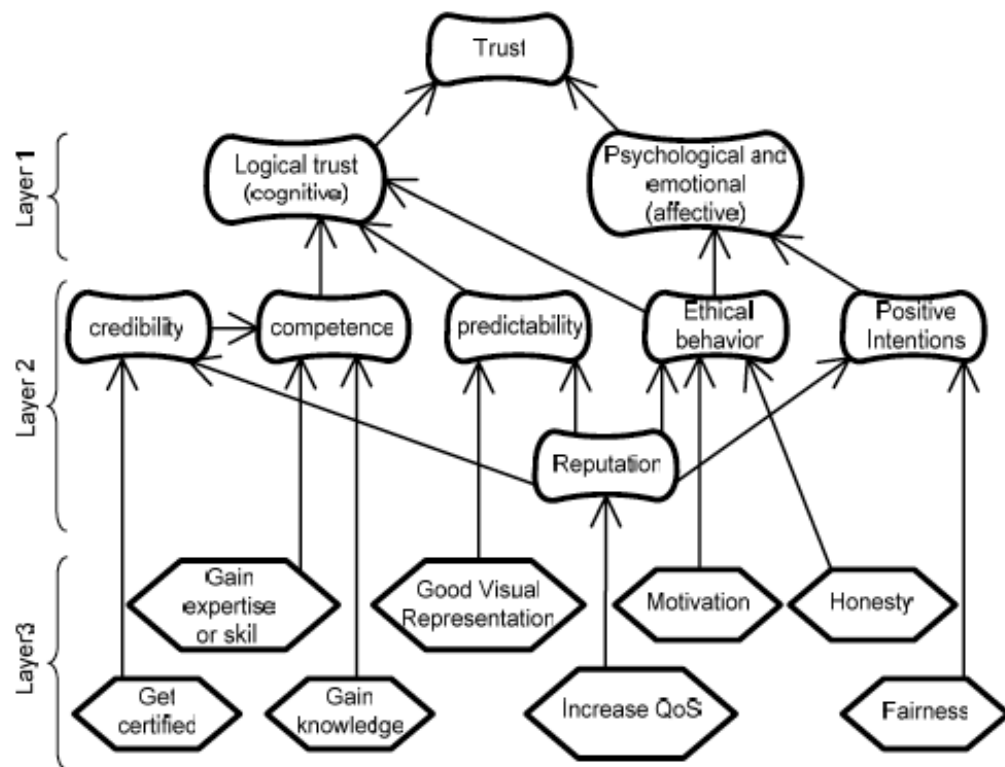


Figure 2.9: Trust reference model

Similarly, the trust analysis is carried out from the viewpoints of the users and not from the perspective of the developer. Also, even though this approach contains information regarding the specific properties of trustworthiness, such as competence, motivation, and prediction, it fails to provide constructive techniques on how to identify requirements for the system to enable it to express the trustworthiness of the trustee.

This line of research although is referred as trust management, however little

relation has with the actual management of trust but more with the management of access control (Cofta, 2007). As trust management we consider the area that aims to provide the means to one entity of a system to assess the trustworthiness of another entity of the system and be able to make a trust justified trust decision. In other words, trust management should aim at systems the enable trust to be communicated among users in order to determine the trustworthiness of a remote user through computer mediated communication and collaboration. At the same time, trustworthy users need the means to enable them to reliably report their true trustworthiness and be recognised as such (Jøsang, Keser, and Dimitrakos, 2005; Cofta, 2007).

### **2.6.5 Human Computer Interaction in the context of trust**

There are two lines of research in this area. On one hand is the research that tries to gain users trust for a system that may or may not be trustworthy by making changes at a human computer interaction level. We believe this approach eventually will fail, as the user sooner or later will recognise if the system cannot be trusted and will create great disappointment to the users. On the other hand, there is research at a Human Computer Interaction level, which aims to identify the technological means to signal signs of trustworthiness to the users. We believe that this category of research adheres to the vision of creating systems that provide the means so that users can assess the trustworthiness of a remote user and proceed to a correct trust decision.

To this end, there has been initial research from Riegelsberger, Sasse, and McCarthy (2005) and Cofta (2007). Riegelsberger, Sasse, and McCarthy (2005) argue that contextual properties will be of higher importance in first interactions and one off encounters, while intrinsic properties of the trustee, such as ability, norm-compliance, and benevolence, are more important in continued exchanges and become increasingly important as trust matures. They also identified two types of signals, symbols and symptoms. Symptoms are signals of trustworthiness that are given as by-product of behaviour and they are preferable to symbols, which may be costly to emit and less reliable.

Similarly, Cofta (2007) identifies three types of evidence that need to be expressed in digital terms in order to enable the assessment of a user's trustworthiness. The first two are competence and motivation, which are considered as easier to be translated into the digital domain, while the third evidence which is competence and it is the hardest one to be translated into the digital domain. More specifically, for competence the technical protocols that can deliver evidence of trust are performance

indicators, for motivation inter-dependent processing and for continuity standards and norms.

This area although it is important for a trustworthy system to be able to signal that trustworthiness to they users, it is outside of the scope of this thesis. We are interested in developing a system that is trustworthy. The case that a system should be able to express such trustworthiness is not considered. Trust is very subjective and it may be impossible to gain every users' trust. However, by making the system trustworthy we hope that the user when she will interact with it she will recognise it and built trust to the system.

### **2.6.6 Trusted Computing**

Trusted computing (Pearson and Balacheff, 2003) is an initiative that was started by the Trusted Computing Group (TCG) (TCG14). TCG consists of AMD, Cisco, Fujitsu, Hewlett-Packard, IBM, Infineon, Intel, Juniper, Lenovo, Microsoft, and Wave. They have proposed a trusted computing platform solution based on tamper-resistant hardware physically located inside the platform. This tamper-resistant hardware provides the computer platform with a root of trust, and it supports an important feature, called integrity challenge of the platform. The integrity challenge feature helps to build a chain of trust, which allows local and remote users to verify whether selected functions and resources of the computing platform have been installed and are operating in a way that satisfies them.

However, trusted computing entails blind trust in the hardware component that acts as the root of trust. This implies that the piece of hardware is being trusted independently of whether it is trustworthy. Apart from technical issues, there are also a lot of concerns about the privacy and freedom of actions of the users who are using trusted computing platforms. Finally, there is a concern from Anderson that whoever controls Trusted Computing infrastructure will acquire a huge amount of power, as a company could have the encryption keys of your word documents.

Despite the limitations and the concerns, this line of research can contribute towards more secure and dependable, and eventually more trustworthy software. However, the achievement of goals of an information system is not only a matter of the software-to-be component of an information system, but the goals are achieved through the collaboration of the software-to-be component and the rest of the components in the information system environment, human or technical. Thus Trusted Computing directly related with the research of this thesis and not discussed further.

### 2.6.7 Computational trust

Another direction of related work is in the area of multi-agent systems. In such systems artificial agents have to make decisions about several issues in various scenarios. Such decisions might be to share information with another agent or not, to accept help or not and from whom to accept, from whom to buy a product and at which price, and so on. As in any decision that contains risk, trust plays an important role in such decisions and is part of the decision making process. Therefore, computational models of trust have been developed that can be used by artificial agents in order to reason about trust. Introducing a way for the artificial agents to reason about trust gives them more solid footing in the human societies into which they are introduced.

One of the earliest computational models is by Marsh (1994), who takes into direct interaction and defines three types of trust, basic trust, general trust, and situational trust. These three values enable the agent to decide whether she will trust another agent. Another approach is by Schillo, Funk, and Rovatsos (2000) where an agent will trust another agent depending on the probability that the second agent is honest in the next interaction. The value is derived from direct interaction information and from information gathered from third party agents that have interacted with the second agent in the past. A cognitive computational trust model was proposed by Castelfranchi and Falcone (1998), where an agent's decision to trust or not is based on competence, dependence, and disposition beliefs about the trustee agent.

However, as Marsh and Briggs (2009) points out "In much the same way that Artificial Intelligence is not real intelligence, the computational concept of trust isn't really trust at all". Although, the thesis and this line of work share the similar principles regarding the trust decision process, this line of work on trust is outside of the scope of this thesis, but the interested reader can see a detailed survey on computational trust in (Sabater and Sierra, 2005).

## 2.7 Chapter summary

This chapter aimed to establish a common language for the understanding of the next chapters as the concept of trust and trustworthiness is overloaded with multiple meanings. To this end, we have reviewed and discussed several definitions of trust and trustworthiness in the human context and we clarified the definitions that we adopt in this thesis (section 2.1). We also discussed trust and trustworthiness in the context of information systems and we clarified the definition of trustworthy



information system (sections 2.2, 2.3). We then discussed high level requirements of methodologies (section 2.4) and evaluation methods (section 2.5).

Moreover, we have reviewed the state of the art of several areas related to this thesis, such as trust modelling (section 2.6.1), trust management (section 2.6.4), security engineering (section 2.6.2), goal satisfaction reasoning (section 2.6.3), Human Computer Interaction (section 2.6.5), Trusted Computing (section 2.6.6), and computational models of trust (section 2.6.7). The main contribution of this chapter is to investigate existing approaches to determine if they can be used or adapted in our approach and show how existing approaches are inadequate to build trustworthy information systems. Most of the approaches ignore the behaviour of the human components, and focus on technical components only. Also, they do not contain the required trust and control software engineering abstractions that can enable developers to reason about trust relationships in a structured way and identify trustworthiness requirements. Table 2.2 depicts the various approaches that were examined in this section along with the issues that they are dealing with. Most of the approaches enable the identification of assumptions, and trust relationships and their reasoning. However, only Secure Tropos (Trento) enables the developer to identify observation functionalities. The state of the art lacks techniques that enable the developer to identify indirect trust relationships, trustworthiness requirements including both observation and deterrence functionalities and the assessment of the system trustworthiness. These limitations are addressed by this work. In the next sections we describe the trust and control constructs of our modelling language and a systematic process to reason about trust relationships and identify and analyse trustworthiness requirements.

Figure 2.10 depicts the area of application of the state of the art in trust engineering and the area of application of JTrust. Trust management deals with the management of trust in the relationship between components of an information system, while trust modelling and HCI approaches are focusing on the relationship between the human and technical components of an information system with the technical system under development. Trusted Computing and goal satisfaction approaches focus on the technical system-to-be and particularly how to make it more trustworthy while neglecting the other components of the system. Computational trust provides mechanisms for a technical component to reason about trust. JTrust though is focusing on the relationships of the developer with the human and technical components of the system and also on the technical system-to-be by identifying trustworthiness requirements.

Table 2.2: Comparison table of state of the art in trust engineering

	Assumptions	Trust relationships	Indirect trust relationships	Trust relationship reasoning	Trustworthiness requirement		Goal reasoning	Trustworthiness assessment
					Obsrv	Deter		
Authors								
i*-trust		✓		✓			✓	
Secure Tropos Trento		✓			✓			
Bimrah		✓		✓				
Yan	✓	✓		✓				
Presti		✓						
Gorski	✓			✓				
Haley	✓							
Elahi		✓		✓				
Trustworthy Computing								
Secure Tropos- trust	✓	✓	✓	✓				
KAOS	✓						✓	
Security problem frames	✓							
CORAS	✓							
UMLsec	✓							
NFR							✓	
Giorgini							✓	
Letier							✓	
Yu							✓	
Lamsweerde							✓	

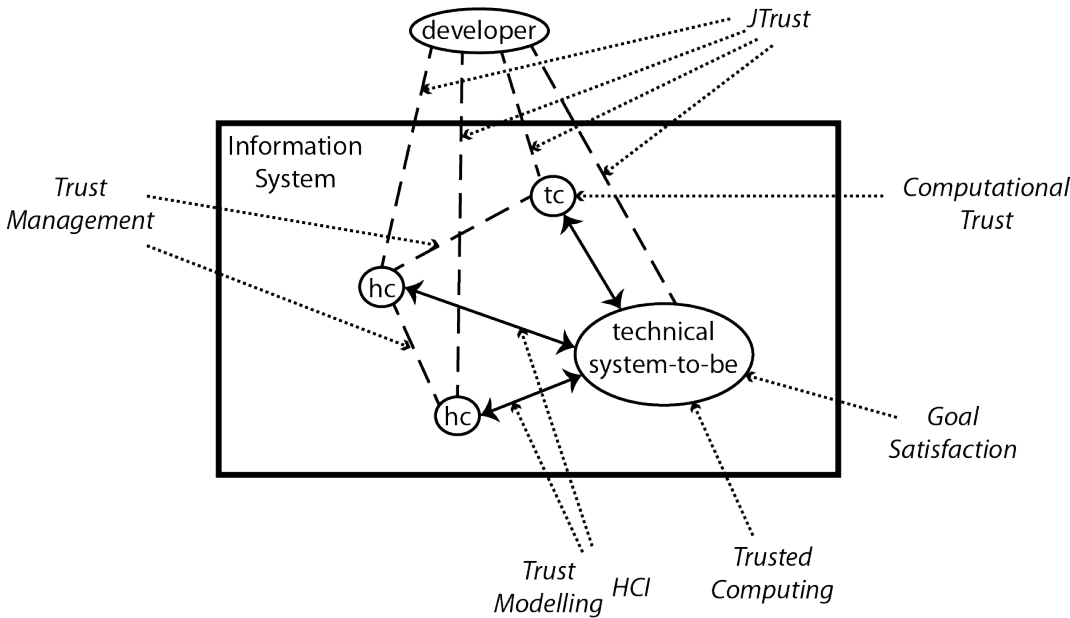


Figure 2.10: Application areas of state of the art in trust engineering

Part II

**JTrust: A Trustworthy  
Information System Development  
Methodology**

## Chapter 3

# JTrust modelling language

*Without recognizing all the entities and their trust relationships  
in a software system during the requirements phase of a project,  
that project is doomed from the start*

---

(Viega, Kohno, and Potter, 2001)

In the previous chapter we carried out a literature review of the areas we believe are relevant to this research. In this chapter, we first describe the requirements for our proposed methodology. These requirements were derived from the literature review and the limitations of the state of the art in trust engineering. In section 3.2 we describe the structure of the methodology. In particular, we explain the components of the methodology, how are related, who are the users, and so on. In section 3.3, we describe a running example, which is used throughout the thesis to better explain the methodology and the benefits that it offers.

Secondly, in this chapter, we propose a trust modelling language to capture the trust relationships between the developer of a system and the entities with which the system is interacting, and also a trustworthiness model to assess the system trustworthiness at a requirements level. As we have introduced in Chapter 1, such trust relationships are critical for the ability of the system to be trustworthy, and may require extra functionality from the system in order to meet its requirements. Such trust relationships need to become explicit and be reasoned about by the developer. We explain the principles our modelling language is based on, give an overview of the Goal Oriented Requirements Engineering (GORE) approach, define the basic terms used throughout the thesis, and we propose a modelling language:

- We describe the constructs related to trust and control that are required by developers to reason about trust relationships and analyse trustworthiness requirements across different projects.

- We propose a meta-model of our trust modelling language, which shows the relationships between our defined constructs.

In other words, in this chapter we attempt to answer research questions 1 and 2, stated in chapter 1: What are the required concepts and their relationships regarding trust that will allow the development of trustworthy information systems at an early stage? In the last section of this chapter we propose a trustworthiness model.

### 3.1 Methodology requirements

Based on the literature review on development methodologies and the identification of limitations of the state of the art in trust engineering certain requirements for the JTrust methodology were derived and are the following:

- Modelling of trust relationships. The methodology must enable the developer to identify and model trust relationships between her and the components of an information system. It should be a system way in order to consider all trust relationships and avoiding omitting trust relationships that are not so obvious, especially the indirect trust relationships.
- Reasoning about trust relationships. The methodology must provide the means to the developer to reason about trust relationships and describe the trust rationale in a uniform and consistent way.
- Identification of trust assumptions. The methodology must enable the developer to identify assumptions that underlie the system development and can harm the trustworthiness of the system if they are not valid once the system is implemented.
- Modelling of trustworthiness requirements. The methodology must enable the developer to model and analyse trustworthiness requirements. These are requirements that will constitute the system trustworthy and have to be fully satisfied by the system.
- Assessment of system trustworthiness. The methodology must enable the developer to evaluate the trustworthiness of the system under development by considering if the system can achieve goals assign to it and the achievement of the system goals assigned to components of the information system.

### 3.2 Methodology structure

The structure of the methodology is depicted in Figure 3.1. The methodology contains a modelling language, a process, and a supporting tool. In turn, the modelling language includes trust and control abstractions and a meta-model that describes the relationships between those abstractions. The JTrust process has a number of activities that the developer can follow in order to use the trust and control abstractions to model and reason about trust relationships, model and analyse trustworthiness requirements, and assess the trustworthiness of the system under development. In addition, it includes algorithms for the calculation of resolution level, confidence level, and system trustworthiness. The last component of the methodology is the supporting tool that enables the developer to construct the trust model of the system-to-be and it automatically calculates the resolution level, confidence level, and system trustworthiness using the respective algorithms.

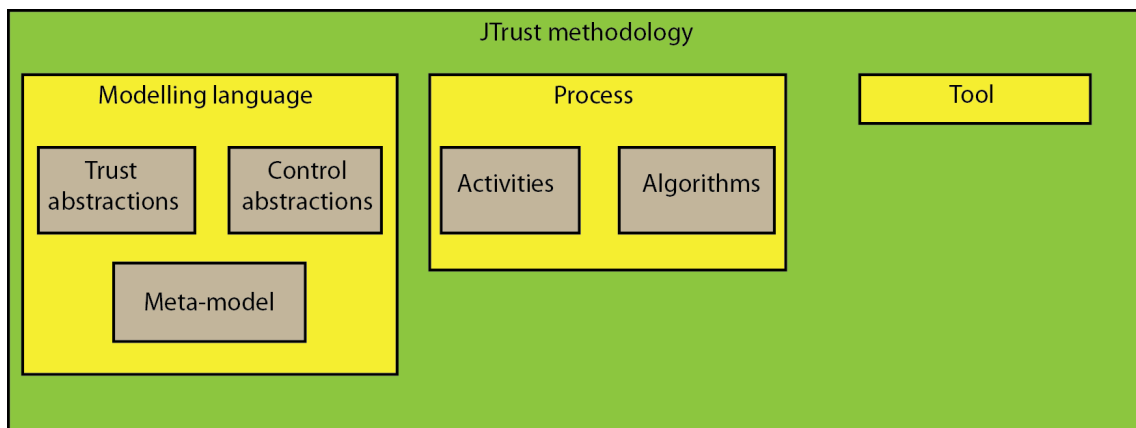


Figure 3.1: JTrust methodology structure

### 3.3 Running example

In this section, we briefly explain an example of an information system and we use it to explain our proposed methodology. We consider a virtual learning environment (VLE) for supporting the education of students at a university. Using such a system the students can download learning material, such as lecture handbooks and slides, communicate, and collaborate with other students. They can be accessed both on and off-campus, meaning that they can support students' learning outside the lecture hall 24 hours a day, seven days a week. In such a scenario one of the goals of the system is to provide lecture slides to the students three days in advance of the lecture. So, students will have the opportunity to go online to view the slides and

be prepared for the lecture. To accomplish this goal the technical system-to-be has another goal to receive the slides from the lecturer. Another goal of the system is to provide the students with access to any relevant administration forms and their student record. Similarly, the administration staff of the university has to upload the administration forms to the system.

Out of this scenario a number of issues arise regarding the trustworthiness of the system. For example, if the lecturer does not upload the lecture slides in advance of the lecture then the system will not be trustworthy because it will not have the lecture slides available for the students. Similarly, if the administration staff do not show the expected behaviour, which is to upload the administration forms to the system, then the system will not be trustworthy, as it does not have the forms required by the students and as a result the students will probably not trust such a system. Whatever method is used by the developer to model the requirements of such a system, it will contain the modelling of the expected behaviour of the lecturer and the administration staff, which are to upload the lecture slides and the administration document. Therefore, there are implied trust relationships formed between the developer and the lecturer and between the developer and the administration staff. If the lecturer and the administrator staff can be trusted to behave as expected then the system will be trustworthy, otherwise if they cannot be trusted it will not be trustworthy. Therefore, the questions that arise from such a scenario are the following:

- How can the developer explicitly capture her trust relationships with the lecturer and the administration staff?
- How can the developer capture any possible indirect trust relationships? For example, there may be a trust relationship between the developer and the administration staff because the administration staff is controlling the lecturer to upload lecture slides in advance of the lecture.
- How can the developer reason about her trust relationships with the lecturer and the administration staff? For example, why is she trusting or not trusting the lecturer or the administration staff.
- How can the developer systematically identify the assumptions that are deriving from her trust relationships? For instance, in case the administration staff controls the lecturer this contains the assumption that the administration staff is trusted to do so.
- If the developer does not trust the lecturer or the administration staff and no



other entity can control them, how can the developer feel confident that they will behave as expected in order to build a trustworthy system?

- How can the developer assess the overall system trustworthiness at a requirements stage in order to decide if she is satisfied with the current level of system trustworthiness and proceed to the next stages of system development?

These are questions from the running example that the JTrust methodology proposed in this thesis envisages to address. Requirements engineering is all about decision making and JTrust envisages to support the developer in making such decisions that will lead to an information system that is as trustworthy as possible.

### **3.4 Confidence as the key to modelling uncertainty**

Modelling is a core process in requirements engineering upon which the implementation of the system is based and to a great extent it will define its success once is put in operation. In requirements modelling the existing system/organisation as well as the possible alternative configurations for the system-to-be are modelled in order to understand complex, real world systems and their behaviour. More particularly, modelling enables the requirements engineer to look at the domain systematically and helps him look beyond less important details and focus on more important parts of the system and reveal key problems in a timely manner. Requirements engineering models serve as basic common interfaces among developers to the various activities of requirements engineering (Van Lamsweerde, 2000). Besides, models help communicate requirements to customers, and other possible stakeholders. Models, also provide a basis for requirements documentation and evolution, and allow for requirements reuse within the domain. An additional benefit of modelling is also to support heuristic, qualitative or formal reasoning schemes during requirements engineering (Lamsweerde, 2001). While informal models are analysed by humans, formal models of system requirements allow for precise analysis by both software tools and humans. The analysis can reveal the presence of inconsistencies in the models, which is indicative of conflicting and/or infeasible requirements.

An information system contains active components, such as humans, devices, and software. Active components have choice of behaviour opposed to passive ones that do not have (Lamsweerde, 2001). Especially humans who are entities with consciousness and make their own decisions. Moreover, economics dictates the use of commercial off the shelf components wherever possible, so developers have neither control nor detailed information about many of their systems' components (Schneider, Bellovin, and Inouye, 1999), which create an uncertainty about their future

behaviour. This situation is well described in (Iivari and Hirschheim, 1996) who contrast design optimism with design pessimism. Design optimism assumes that only existing resource constraints limit the development and acquisition of desired information systems, and that the systems can be implemented without difficulty. Design pessimism, in contrast, assumes no such ease. People are assumed not to behave rationally and the social nature of information systems development makes any design exercise difficult.

As a result modelling includes modelling of active components that their behaviour can be unpredictable and different from the one in the model. Consequently this will have negative implications for the operation of the system to be. More particularly, Goals have long been recognised to be essential elements involved in requirements modelling. Unlike requirements, a goal may in general require the co-operation of a number of multiple components to be achieved (Lamsweerde, 2001). A goal under responsibility of a single component in the software-to-be becomes a requirement whereas a goal under responsibility of a single component in the environment of the software-to-be becomes an assumption (Van Lamsweerde, Darimont, and Letier, 1998; Van Lamsweerde and Willemet, 1998). A goal assigned to especially an active component is an assumption that the active component will behave according to the model. Important distinction must be made about requirements and environment assumptions (sometimes called expectations). Even though they are both optative, requirements are to be enforced by the software, while assumptions are to be enforced by active components in the environment. Unlike requirements, assumptions cannot be enforced by the software-to-be, but they will hopefully be satisfied thanks to organisational norms, and regulations, physical laws, etc. (Lamsweerde, 2001). The assumptions specify what the system expects of its environment (Jackson, 1997; Parnas and Madey, 1995) and consequently incorrect assumptions about the environment of a software system are known to be responsible for many errors in requirements specifications (Lamsweerde and Letier, 2004).

Behind the approaches that are used to develop information systems, lie a number of implicit and explicit assumptions and views. Although, alternative assumptions and views guide the information systems developer in the choice of various analysis, design and implementation options and hence have important consequences for the development of successful system, only rarely they appear to be critically reflected upon or challenged (Iivari and Hirschheim, 1996). First-sketch specifications of requirements, goals, and assumptions in requirements modelling are often too ideal and they are likely to be violated from time to time in the running system due to unexpected behaviour those active components. A survey showed only 10% of these failures are due to technical issues, with 90% attributed to social and organ-

isational factors (Doherty and King, 1998). The lack of anticipation of exceptional behaviours may result in unrealistic, unachievable and/or incomplete requirements (Lamsweerde, 2001). Also, developers face uncertainty in prediction of the impact of their interventions due to complex interactions between the elements in the design and the change context (Lyytinen and Newman, 2008).

This situation requires information systems to be viewed from a different perspective. An information system at the organisational level may be considered a technical artefact or tool, just as a hammer, for example, which is socially produced and used, but which does not embody any deeper social meaning. Such a view is technical. Alternatively, an information system may be considered an artefact, which entails inherent social and organisational aspects. This we characterise as a social view. An intermediate position between these two extremes, termed the socio-technical view, emphasises that an information system comprises both a technical subsystem and a social subsystem that should be designed jointly (Iivari and Hirschheim, 1996; Bostrom and Heinen, 1977). Nevertheless, the prominent tendency in software modelling until some years ago was to abstract programming constructs up to requirements level rather than propagate requirements abstractions down to programming level (Mylopoulos, Chung, and Yu, 1999). In others words the social aspects regarding the behaviour of the active components are not taken into account sufficiently who are endangering the fulfilment of the goals of an information system. Even though a sociotechnical view of information systems has been adopted recently, the state of the art still fails to provide abstract constructs, which are applicable over time and space, and enable the developer to feel confident in the assumptions that are underlying the system development.

In our approach, we adopt a sociotechnical view of an information system, a system that combines a technical system and its environment. The environment can be human components or other existing technical components that will interact with technical system-to-be. Moreover, Information technology has the potential to change social and organisational structures and simultaneously be affected by these structures in its design, implementation, and use (Luna-Reyes et al., 2005). Therefore, it is not sufficient simply to assemble components that are themselves trustworthy. Integrating the components and understanding how the trustworthiness dimensions interact is a central challenge in building a trustworthy information system (Schneider, Bellovin, and Inouye, 1999).

The assumptions underlying the modelling of an information system originate because of the dependencies of the developers on the system components. When a developer is modelling an information system, she depends on the active components of that system to behave as modelled in order the system fulfil its goals. The

dependencies of the developers are eventually becoming dependencies of the technical system-to-be once put in operation and potential vulnerabilities if the active components do not behave as modelled. Thus, the focus should be shifted on how to remove the potential vulnerability and uncertainty a dependency is bringing. The developer by some means should be confident that when modelling the behaviour of an active component, this component will behave as modelled once the system is put in operation.

The developer when modelling an information system that contains active components requires confidence. Confidence in her decisions during requirements modelling, i.e., confidence in the dependencies on the active components that they will behave as modelled and confidence that her assumptions are valid, confidence in order to remove any uncertainty about the future behaviour of the active components and the potential vulnerabilities they may introduce to the system. In this thesis we view confidence as an optimisation of the decision making process during requirements engineering as it provides the reassurance about the behaviour of the active components.

To this end, we adopt the model of confidence (Figure 3.2 from (Cofta, 2007)). The consciousness of confidence originates from two separate sources: trust and control (Das and Teng, 1998; Cofta, 2007). If individuals are going to depend on something and they want to feel confident in that, there are two ways to achieve that confidence, either by trust or control (Cofta, 2007; Lewicki and Bunker, 1996; Das and Teng, 1998). The confidence of an individual in another individual depends upon her trustworthy behaviour. The cause of that behaviour can be her trustworthiness, which enables the first individual to trust her. However, if the trustworthiness is low then control can be applied in order to change the behaviour of that individual and cause her to behave in a trustworthy manner. The first individual cannot trust her but she can still feel confident in her behaviour because of control. Therefore, control means that there is a way by which a component of a system will behave in a predictable way and no unpleasant surprises will occur. Effective control generates a sense of confidence (Cofta, 2007; Lewicki and Bunker, 1996; Das and Teng, 1998). On the other hand there is trust which means, as we defined it in the previous chapter, the positive expectations about the behaviour of an active component by which there can be positive or negative affection. Trust generates confidence since the trustor has a positive stance towards the active component's behaviour even in the absence of control. Particularly, trust is required when there is lack of control (Dasgupta, 2000) in order to gain confidence. In contrast, when it is fully possible to trust an active component then there is no need to control its behaviour (Das and Teng, 1998). Therefore, during requirements engineering the developers need

to use means of trust or control in order to feel confident in her dependencies on active components and her assumptions about their behaviour.

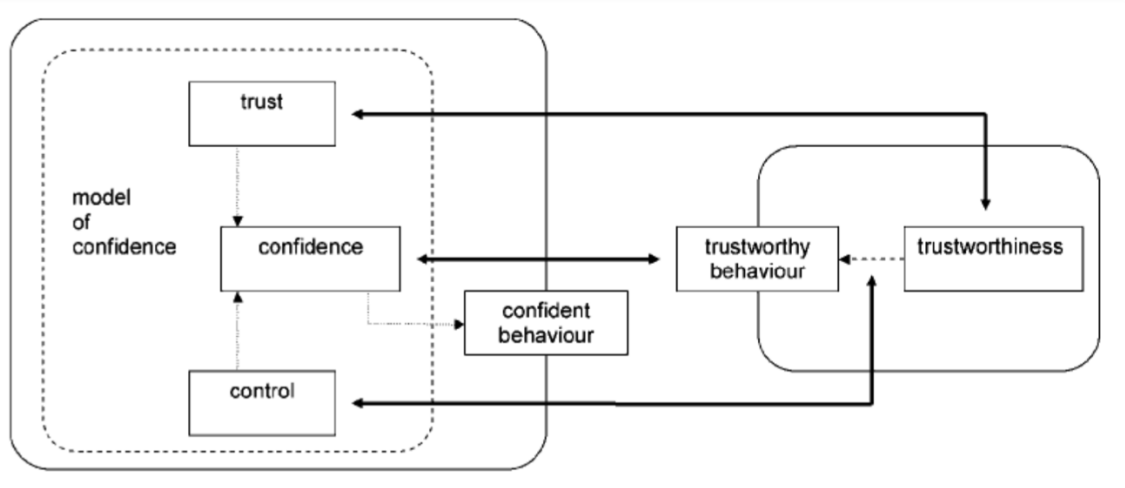


Figure 3.2: Model of confidence

However, there is a duality of trust and control and not dualism (Möllering, 2005). These concepts are not opposite and cannot be investigated independently. Control is presupposed in case of trust and trust is presupposed in case of control. When there is trust in the behaviour there is the assumption that this behaviour has a meaning and a connection in the environment. When there is control of the behaviour by the environment there is the assumption that the environment is trusted to do so. Also, trust refers to control and vice versa. Whether there is trust or not, there is also an assessment of how much trust is needed because of possible existence of control. Whether to control or not, there is also an assessment of how much trust there is already. In addition trust creates control, and control creates trust. Controls force components to behave in a specific way and this can create trust in their behaviour. Also, inside a particular environment trust can be generalised according to an environment norm and this trust creates control on components. For example, when components know that their trustworthiness is assumed in a particular environment then this will make them feel obligated to behave in a specific way. Components refer to their environment in order to understand which behaviour is expected and which behaviour is not expected. The developer should consider all these issues when reasoning about his trust relationships with the active components of the system-to-be and evaluate whether control is required on top of trust or vice versa. In such cases there is also a trade off between the extra confidence and the less complexity added to the system because of control.

The ultimate goal of a trust analysis is to support the developer in considering

the aforementioned issues. To this end appropriate software engineering abstractions related to trust and control should be developed and accompanied constructive techniques that can guide the developer in using the abstractions. Also, it is necessary the developer to understand the relationships among the entities within the socio-technical system; in particular, to know the dependencies among the entities and how they are resolved, i.e., dependencies that will be fulfilled once the system is put in operation and the assumptions that are made are valid. Dependencies thus need to be identified and resolved at a requirements stage in order to produce robust requirements and hence more trustworthy system.

Traditional system modelling approaches though treat requirements as consisting only of processes and data and do not capture the rationale for the software systems, thus making it difficult to understand requirements with respect to some high-level concerns in the problem domain. Also, most techniques focus on modelling and specification of the software alone. Therefore, they lack support for reasoning about the composite system comprise of the system-to-be and its environment (Lamsweerde and Letier, 2004). Non- functional requirements are also in general left outside of requirements specifications. Additionally, traditional modelling and analysis techniques do not allow alternative system configurations where more or less functionality is automated or different assignments of responsibility are explored, etc. to be represented and compared. The prominent tendency in software modelling has been to abstract programming constructs up to requirements level rather than propagate requirements abstractions down to programming level (Mylopoulos, Chung, and Yu, 1999). This explains why the stakeholders with their needs and the rest of the social context for the system could not be adequately captured by requirements models.

As mentioned in Chapter 1 the first two objectives of this thesis is to define a requirements based modelling language and methods for developing trustworthy information systems. In the foundations of our methodology is the Goal Oriented Requirements Engineering (GORE) approach, which is concerned with the use of goals for eliciting, elaborating, structuring, specifying, analysing, negotiating, documenting, and modifying requirements (Lamsweerde, 2001). GORE has emerged as a prominent approach within RE and the main concept in GORE is goals. Goals can be formulated at different levels of abstraction ranging from high-level, strategic concerns to low-level, technical concerns (Lamsweerde, 2001). Goals also cover different types of concerns, such as functional concerns associated with the service to be provided, and non-functional concerns associated with quality of services, i.e., safety, security, accuracy, performance, and so forth (Lamsweerde, 2001). Goals are important to requirements engineering for many reasons:

- Goals enable the achievement of requirements completeness. They provide the criterion for sufficient completeness of requirements specification. The requirements specification is complete if all goals can be achieved (Yue, 1987).
- The use of goal enables to avoid the inclusion of irrelevant goals. A requirement is pertinent with respect to a set of goals if its specification is used in the proof of one goal at least (Yue, 1987). Therefore, a system functionality is relevant if it is part of reaching a goal of the stakeholders, otherwise its implementation is not justified.
- Explaining requirements to stakeholders is another important issue. In particular a goal refinement tree provides traceability links from high level strategic objectives to low level technical requirements, so goals may be used to relate the software-to-be to organisational and business contexts (Yu, 1993). Through this way the stakeholders can get answers to questions such as "why do we need to do this", and "how to we reach this" and so on.
- Goal refinement provides a natural mechanism for structuring complex requirements documents for increased readability (Lamsweerde, 2001). This refinement is occurring by asking how, and how else, and how good each alternative is.
- Requirements engineers are faced with many alternatives to be considered during the requirements elaboration process. Goal refinements provide the right level of abstraction at which decision makers can be involved for validating choices being made or suggesting alternatives overlooked so far (Lamsweerde, 2001).
- Managing conflicts among multiple viewpoints is another problem. Stakeholders are different in their interests and priorities of these interests. This may lead to conflicts about what requirements to fulfil and how, and how well to fulfil them. Goals enable to detect conflicts among requirements and to resolve them eventually (Van Lamsweerde, Darimont, and Letier, 1998). For example achieving, a goal may deny the achievement of another.
- Separating stable from volatile information is another important concern for managing requirements evolution. Requirements are more likely to evolve rather than goals. The higher a goal the more stable it will be. Different system versions often share a common set of high-level goals. The current system and the system-to-be correspond to alternative refinements of common

goals in the goal refinement graph and can therefore be integrated into one single goal model (Lamsweerde, 2001).

- Goals drive the identification of requirements to support them; they have been shown to be among the basic driving forces, together with scenarios, for a systematic requirements elaboration process (Dardenne, Lamsweerde, and Fickas, 1993).
- Dependencies among components of the domain of interest (Yu, 1995).

In (Lamsweerde, 2001) there is an overview of the GORE field and a report by (Lapouchnian, 2005) provides an overview of some of the approaches that follow the principles of GORE.

### 3.5 JTrust modelling language concepts

The GORE approach is adopted because it already has useful concepts such as actor, goal, and dependency. These concepts are used to describe the system and its environment. We adopt these concepts and we enrich it with trust related concepts including experiential trust, reported trust, normative trust, and external trust, and control related concepts including observation and deterrence. These concepts allow the developer to capture trust relationships and to reason about them in a systematic way. Also, the concepts enable the developer to make a justified decision about the future behaviour of the active components of the information system-to-be at a requirements stage. Moreover, the developer is able to model and analyse trustworthiness requirements and assess the trustworthiness level of the system-to-be. Here we illustrate the main concepts of GORE that will be used in the rest of the thesis.

**Goal** represents a component's strategic interest and is a condition that system components want to achieve (Yu, 1995). Goals capture the reasons why a system is required (Antón and Potts, 1998), thus becoming objectives that the information system should achieve through the cooperation of the components of the system (Van Lamsweerde, 2000). Of course, this can be done in more than one way so alternatives of achieving a goal can be considered. There can be different types of goal (Lamsweerde, 2001). Functional goals underlie services that the system is expected to deliver whereas non-functional goals refer to expected system qualities such as security, safety, performance, usability, flexibility, customisability, interoperability, and so forth. More particularly, there can be satisfaction goals, which are functional goals concerned with satisfying request from system components, information goals



are functional goals concerned with keeping such components informed about object states. Non-functional goals can be specialised in a similar way. For example, accuracy goals are non-functional goals requiring the state of software object to accurately reflect the state of the corresponding monitored/controlled objects in the environment. Such goals are often overlooked in the requirements engineering process and their violation may be responsible for major failures. Performance goals are specialised into time and space performance goals, the former being specialised into response time and throughput goals. Security goals are specialised into confidentiality, integrity, and availability goals. Also, there are privacy goals that are specialised into authentication, authorisation, identification, data protection, anonymity, pseudonymity, unlinkability, and unobservability (Kalloniatis, Kavakli, and Gritzalis, 2008).

In GORE there are two categories of goals; Hard goals or simply goals that have clear-cut definition and clear-cut criteria to judge whether it are satisfied; And goals that are called softgoals and their satisfaction cannot be established in a clear-cut sense. Thus there cannot be trust based reasoning about these goals if we do not have clear criteria. But, these goals can be refined further into goals that have clear criteria and contribute to the satisfaction of softgoals. These goals then can be used for trust analysis.

The identification of goals is not an easy activity, as a lot of goals are implicit. Thus the goal elicitation process should analyse the current system environment and elicit goals from sources such as documents, interviews, scenarios, use cases, policies, and so on. Scenarios help for identifying goals and the different ways through which they may be implemented and fulfilled. An informal technique for finding out more abstract, parent goals is to keep asking "why" questions about operational descriptions already available (Van Lamsweerde, 2000). The usefulness of identifying more abstract goals is that once a more abstract goal is identified, it may be possible to refine it and identify subgoals that were initially not identified. Thus, the identification and refinement of more abstract goals leads to a more complete requirements specification.

Once some goals have been identified the next step is to refine them into progressively simpler goals until these goals can be easily implemented. The refinement of goals is captured by goal models. Goal models serve as means of communication among developers and stakeholders, but also as an abstraction specification of the system-to-be. An informal technique for finding out subgoals and requirements is to keep asking "how" questions about the goals already identified (Van Lamsweerde, 2000), but formal goal refinement patterns may also prove effective when goal specifications are formalised. They help finding out subgoals that were overlooked but

are needed to establish the parent goal (Van Lamsweerde, 2000).

**Decomposition** is the relationship link that that represents the goal refinement and essentially defines the goal structure. The refinement of goals occurs through AND/OR refinements. So, when a goal is refined through and AND-Decomposition, it means that fulfilling all subgoals is mandatory for the fulfilment of the goal that is being refined. When a goal is refined through an OR-Decomposition it means that fulfilling one of these goals is enough for the fulfilment of the goal that is being refined. In other words, the OR-Decomposition represents the alternatives ways to fulfil a goal.

**Actor** is an entity of the domain of interest that possesses strategic goals and intentionality by carrying out actions that will fulfil those goals (Yu, 1995). It is a unit that encapsulates intentionality, rationality, and autonomy. Actors want to achieve goals either using their own capability or depending on another actor. An actor can be a human entity playing a certain role, an organisation, or a technical entity, such as software or device. For example, a lecturer represents a human actor and a university virtual learning environment (VLE) represents a technical actor.

**Dependency** is the relationship link that indicates that one actor depends on another actor to fulfil a goal (Yu, 1995; Yu et al., 2011). The goal around which the dependency centres is named dependum. By depending on another actor for a goal, an actor is able to achieve goals that otherwise he would not be able to achieve alone, or not as easily or as well. Nevertheless, there is no guarantee that the dependee will fulfil the goal and also the dependee is given the freedom to choose how to do it, as the depender is only concerned about the outcome.

A **Corresponds** link represents the relationship between the goal of the depender and the goal of the dependee that satisfies the depender's goal, which is the dependum. The tail element of the link is the depender's goal, while the head element is the goal that the dependee is achieving in order to achieve the goal of the depender.

A natural drawback of a dependency is that the depender becomes vulnerable. If the dependee fails to fulfil the dependency then this will have negative consequences for the depender to achieve his goals. Therefore, the depender is exposed to the behaviour of the dependee, which is not always predictable. Such dependencies need somehow to be resolved in order to be confident that the dependee will fulfil the dependency.

**Trust relationship** is any dependency relationship between the actors of an information system because in dependencies the depender has expectations from the dependee to fulfil the dependency by achieving the goal instead of him. Moreover, the outcome of the dependency can have positive or negative effect on the dependee.

Similarly, the trustor is the depender and the trustee is the dependee. However, in a trust relationship the level of trust might be high or low. For this reason the concept of dependency is used to capture trust relationships. Furthermore, trust relationships can be divided into two categories:

- **Direct trust relationship**, which is a trust relationship that exists for its own reasons.
- **Indirect trust relationship**, which exists only to support other trust relationships.

For such trust relationships, not only is the trustor/depender concerned, but also the developer of an information system. The developer's goal is to build an information system that satisfies the goals of the stakeholders who are modelled as actors in her model along with their dependencies.

**Resolution** is the means to offset the potential vulnerability that a dependency introduces. In order to find resolutions the developer needs to ask the question of why does he feel confident that the dependee will achieve the goal and hence fulfil the dependency. The resolutions essentially reveal the assumptions that must be valid for the system to be built on strong foundations. In the running example, the system has a goal to receive the lectures slides, which cannot accomplish on its own. Therefore, the system depends on the lecturer for uploading the lecture slides. For this dependency and its accompanying potential vulnerability a resolution is required to be identified by the developer in order to build confidence in the fulfilment of the dependency.

A **Resolves** link represents the relationship of the resolution with the dependency that it is resolving. The tail element is the resolution and the head element is the resolved dependency. Based on the confidence model presented in the previous section that states that confidence can be achieved through trust and control, we define two types of resolutions, trust resolution and control resolution. So, in our running example the developer can feel confident that the lecture will upload the slides three days in advance either because she trusts him or she controls him to do so.

A **Trust Resolution** indicates that the developer trusts that the dependee will fulfil the dependency by achieving the goal once the information system-to-be is put in operation. For example, the developer of the VLE can choose to resolve the dependency on the lecturer to upload the lecture slides three days in advance of the lecture with a trust resolution because she trusts the lecturer for doing so. We propose four different types of trust resolution:

- **Experiential Trust** is trust that originates from previous direct experience with the trustee. The parties have verifiable information by first hand interactional or transactional experience with each other (McKnight and Chervany, 2006; Sabater and Sierra, 2005). This information concerns the trustee's intrinsic properties, including ability and motivation, which affect trust more in a relationship that has been established for some time (Riegelsberger, Sasse, and McCarthy, 2005). In the running example, if there is an experiential trust resolution of the dependency on the lecturer to upload the slides then this means that the developer trusts the lecturer to do so on the basis of previous direct experience that she had with the lecturer.
- **Reported Trust** is trust that originates from a third party (the reporter) who reports that the trustee is trustworthy (Sabater and Sierra, 2005; Bigley and Pearce, 1998). This is valuable when the trustor does not have first-hand information of the trustee. In effect a new dependency is introduced; the trustor depends also on the third party for the accuracy of the information that she is providing to her. This is an indirect trust relationship because it only exists to support the original trust relationship. For, example the developer of the VLE may feel confident in the fulfilment of the dependency with the lecturer because of reported trust resolution. This means, for example, that the university administration, which is another actor inside the VLE system environment, reports that the lecturer can be trusted to do so. In effect, then the developer is depending of the university administration for the accuracy of the information. This trust relationship on the university administration is an indirect trust relationship because it supports the original direct trust relationship with the lecturer.
- **Normative Trust** is trust that originates from the system environment norms. Trustees are motivated to behave in a specific way by their desire to act in accordance with internalised norms (Riegelsberger, Sasse, and McCarthy, 2005; Sabater and Sierra, 2005). As mentioned in the previous section trust assumes the existence of control and also it creates control (Möllering, 2005). Environment norms constitute actors obliged to behave in a certain way. Therefore, in a way, actors of a system environment are embossed by the environment norms to show specific behaviour. By system environment we mean the technical system-to-be, the stakeholders, and the humans and other technical systems that interact with the system-to-be. It is the developer's task to set the boundaries of the system environment. Back to our example, the developer may decide to resolve the dependency on the lecturer by using normative

trust resolution. This means that the developer decides to trust the lecturer to behave as modelled, i.e., upload the lectures slides three days in advance of the lecture, because it is an environment norm in the specific university environment for the lecturers to do so.

- **External Trust** is trust that originates from sources outside of the system environment. Such sources can be laws and external policies. In our running example, the developer may decide to resolve the dependency on the lecturer using an external trust resolution. This means the developers trusts the lecturer to fulfil the dependency because there is government policy that says that it is the lecturer's responsibility to upload the slides three days in advance of the lecture.

Trust resolution is one way to resolve the uncertainty that a dependency is introducing. Confidence though in the fulfilment of a dependency can also be achieved through control.

**Control resolution:** Control is a key source of confidence (Das and Teng, 1998; Cofta, 2007). Control is the power that one actor has over another actor to influence her behaviour (Marsh and Briggs, 2009). In essence, when a developer uses a control resolution as a resolution of a dependency this means that she is confident that the dependee will fulfil the dependency by achieving the goal once the system is put in operation, because either the depender or another actor is controlling the dependee and forcing her to do so. However, if the controller is a third party, such as another actor, then this situation implies that there is a new dependency, and eventually a trust relationship, on the third party to control the original dependee. This new dependency is required to be resolved as well. This new trust relationship is an indirect trust relationship. In our example, the developer of the VLE may decide to resolve the dependency on the lecturer and feel confident in the fulfilment of the dependency once the system is put in operation, with a control resolution because the university administration is controlling the lecturer and forcing him to do so. Therefore, there is a new dependency on the university administration to control the lecturer for uploading the lecture slides.

An **Introduces** link represents the relationship between a reported trust resolution or a control resolution and the new dependency they are introducing. The tail element is a reported trust resolution or a control resolution and the head element is the new dependency. The new dependency in case of reported trust resolution or control resolution is on the reporter or on the controller respectively.

Control requires knowledge and influence from the depender's side. Knowledge specifies the ability of an actor to gather information about another actor's be-

haviour in order to decide whether to execute an action that will directly influence her behaviour. In addition, control specifies the action that is required for the dependee to behave in an expected way. In other words, control is in place in order to punish the depender in case of deceitful behaviour (Mayer, Davis, and Schoorman, 1995). We define two concepts that represent the above notions, observation and deterrence:

- **Observation** is the continuous examination of whether that dependee actor is behaving as expected and she is fulfilling a dependency. It is required as part of an effective control mechanism in order to alert that a dependency is not being fulfilled and possibly to trigger further actions. Back in the running example, the observation capability apparently may be the university administration to check three days in advance of a lecture, if the lecture slides have been uploaded by the lecturer.
- **Deterrence** is the prevention of a dependee's goal in case she fails to fulfil a dependency, so that it can influence her behaviour. Only through observation a control mechanism cannot be effective. It requires also an action that can be taken against a dependee who fails to achieve a goal of a dependency. This action should prevent the dependee from accomplishing one of his own goals, so eventually it will influence her behaviour. The more important is the dependency for the depender the more important the goal that is prevented should be. Deterrence acts both as a threat and as a punishment for the dependee. In the running example, the deterrence may be that the lecturer will be given a fine and it will be deducted from her salary if she does not upload the slides three days in advance.

A **Prevents** link shows the relationship between the deterrence and the goal of the dependee that it is being prevented. The tail element is the deterrence, while the head element is the prevented goal.

A **Contributes** link represents the relationship between the observation and the control resolution and between the deterrence and control resolution. The tail element is observation or deterrence and the head element is control resolution. By modelling the resolutions of the dependencies of the information system, the developer is reasoning about the trust relationships that exist inside the information system-to-be. This leads to the explicit exposure of the trust assumptions that are underlying the development of an information system and which we define them as entailments.

**Entailment** is a condition of trust that is required to be valid for having con-

confidence in the dependency from which it is originally generated. Entailments are created because of the resolutions that have been defined by the developer. In our running example, if there is a control resolution of the dependency on the lecturer because the university administration is controlling the lecturer to upload the lecture slides, then this resolution requires an entailment that the university is trusted to control the lecturer. Similarly, if there is a normative trust resolution of the dependency on the lecturer because it is a university environment norm for the lecturers to do so, then this resolution requires an entailment that the university environment norm is trusted for the lecturer to upload the slides three days in advance. An entailment has to be validated against the reality.

A **Requires** link represents the relationship between the resolution and the required entailment. The tail element is the resolution and the head element is the entailment. We define this relationship as "Requires" because the entailments are required to be valid in order the development of the system-to-be to be based on sound trust assumptions. Such assumptions of trust conditions need evidence in order to be justified and considered valid so as the development of the information system to be based on strong foundations and the system to be trustworthy once put in operation. For example, if there is an entailment that the university environment norm is trusted, is there any evidence that supports this assumption? If such assumptions are left unquestioned then they may introduce vulnerabilities to the system and the system will not be trustworthy.

The assessment of the entailments is a complex task. The developer needs to seek any kind of evidence that will enable him to decide whether the entailment is valid or not. Sources of evidence for the developer to look into are historical data, surveys, interviews and so on. If there is evidence that supports the validity of the trust assumption then the developer can be confident that the dependency will be fulfilled once the system is put in operation. In contrast, if there is no evidence that supports the validity of the entailment then this dependency might be a potential vulnerability that will affect the ability of the system to meet its goals.

### **3.6 Trustworthiness requirements**

The goals of an information system are accomplished with the collaboration of the actors that are part of that information system. In order for the goals to be achieved every actor should do its part. The technical system-to-be has to meet its requirements and the other technical components and human actors need to behave in the way they were modelled during the requirements modelling stage of system development. We have proposed a number of concepts that enable the developer

to reason about the trust relationships with the human actors or other technical actors. Nevertheless, there will be cases where human or other technical actors are not trusted or cannot be controlled by another human actor to behave in the way that it is desired. This can lead to potential vulnerabilities and to a system that will not be able to achieve its goals. For this reason, we recognise the need of requirements that will influence the actors to behave in a specific way. We call such requirements trustworthiness requirements.

This must be a requirement of the technical system-to-be, since the developer has direct control over the software that she is developing. The software-to-be can act as an extension of the developer and compel the other actors to behave in a desired way once is put in operation. We therefore define trustworthiness requirement as follows:

*Trustworthiness requirement is the functionality of the technical system-to-be that influences the behaviour of a dependee actor, with respect to the fulfilment of a dependency inside an information system.*

This functionality must provide the system with the ability to control other actors. As discussed in the previous section control consists of observation and deterrence capabilities. Consequently, the system requires having at least one observation and at least one deterrence capability, in order the control to be effective in influencing a dependee actor to fulfil a dependency inside the information system. The observation capability is in place to check whether a dependee actor fulfils a dependency. The deterrence capability is in place in order to prevent the dependee actor to achieve one of his own goals. The more important to the system trustworthiness is a dependency, the more important the goal of the dependee the developer can choose to prevent through the deterrence.

### **3.7 Meta-model of the JTrust modelling language**

In section 3.5 the trust and control abstractions of our modelling language were defined and motivated. In this section we present and describe the meta-model of our modelling language (Figure 3.3) and we justify the relationships between the concepts. The meta-model was developed using standard UML notation. Therefore, the rectangles represent classes while the arrows represent associations and generalisations. We illustrate each aspect of the meta-model using the running example describe in section 3.3

In the meta-model the actor is the main concept and has zero to many goals that she wants to achieve. There is a hierarchy of the goals and the high level goals are



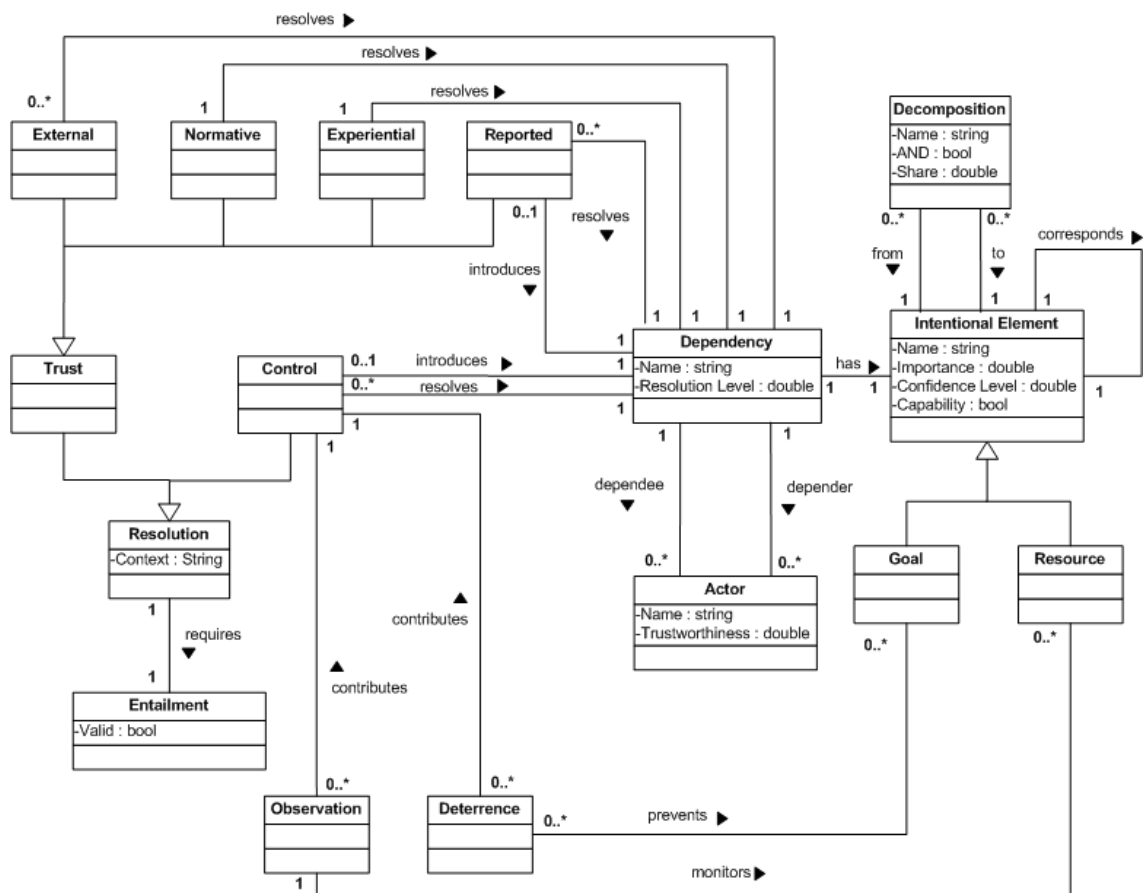


Figure 3.3: JTrust modelling language meta-model

decomposed into smaller goals through AND/OR decomposition links. Figure 3.4 shows the goal structure of the VLE technical system and how the goals are linked together. The high level goal of the VLE system is to provide services, which is decomposed into smaller and more concrete goals. Two subgoals of the system-to-be are to provide administrative services and academic services. These are further decomposed. A subgoal of the academic services is to manage the lecture slides, which is further AND-decomposed to obtain lecture slides from the lecturer and provide the lecture slides to the student. In addition, the goal to manage timetable is AND-decomposed to obtain time and provide timetable

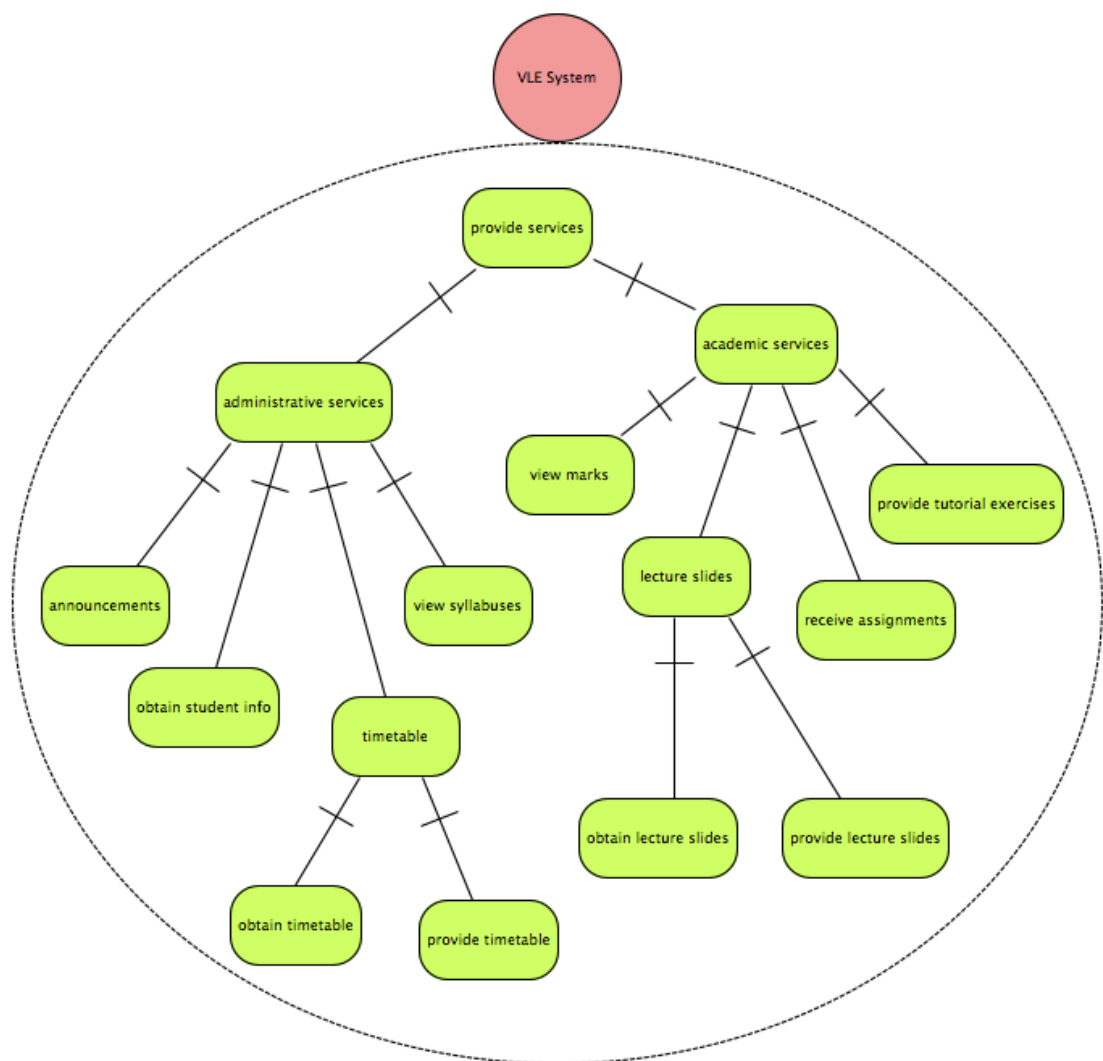


Figure 3.4: Goal model example

For some of these goals the actor is able to achieve them by herself. However, there are goals that cannot achieve by her and she therefore needs to depend on other

actors to achieve them for her. There is a one to one correspondence relationships between the goal that cannot be achieved by the actor herself and the corresponding goal of the dependee actor. As a result she forms dependency relationships with zero to many actors where she is the depender. At the same time, other actors might depend on her in order to achieve goals for them. So the initial actor can also be a dependee in zero to many dependencies. Figure 3.5 illustrates how these concepts come together. In the VLE scenario there are a number of actors involved, such as the VLE technical system-to-be, the lecturer, the university administration, and the student, that for the goals that cannot achieve by themselves they depend on other actors to do so. For example, the system has a goal to obtain the timetable so there is a correspondence link between the obtain timetable goal and the upload timetable goal of the university administration. Similarly, the student has a goal to submit assignments so there is a correspondence link between the submit assignment goal and receive assignment goal of the VLE System. The VLE system has a goal to obtain the lecture slides, so there is a correspondence link between the obtain slides goal and the upload slides goal of the lecturer. The correspondences links result into dependencies between the actors.

In order to have confidence in the fulfilment of a dependency, by the dependee, the dependency needs to be resolved by trust or control. In case of trust there are four different types of trust. A dependency can have zero to many reported or external trust resolutions, depending on how many are the reporters or the external sources of trust. Also, a dependency can have zero to one experiential or normative trust resolutions, as there can be only one norm and the depender is one individual. Furthermore, a dependency can have zero to many control resolutions, depending on how many are the controllers. A reported trust resolution introduces one dependency, on the reporter, while a control resolution introduces zero to one dependency depending on whether the controller is the depender herself or a third party. If the controller is the depender then there is no new dependency, otherwise if the controller is a third party then there is new dependency on the third party. Dependencies that are introduced because of reported trust or control resolutions need to be resolved again in the same way. Figure 3.6 illustrates how these concepts come together. The dependency on the lecturer to upload the lecture slides is resolved with control, where the university administration is the controller. This means that there is the initial belief after discussion with the stakeholders that the administration is controlling the lecturer in order to upload the lecture slides. As a result, this introduces a new dependency on the university administration to control the lecturer because if it does not do so then the lecturer will not upload the lecture slides. The new dependency on the administration is resolved with expe-

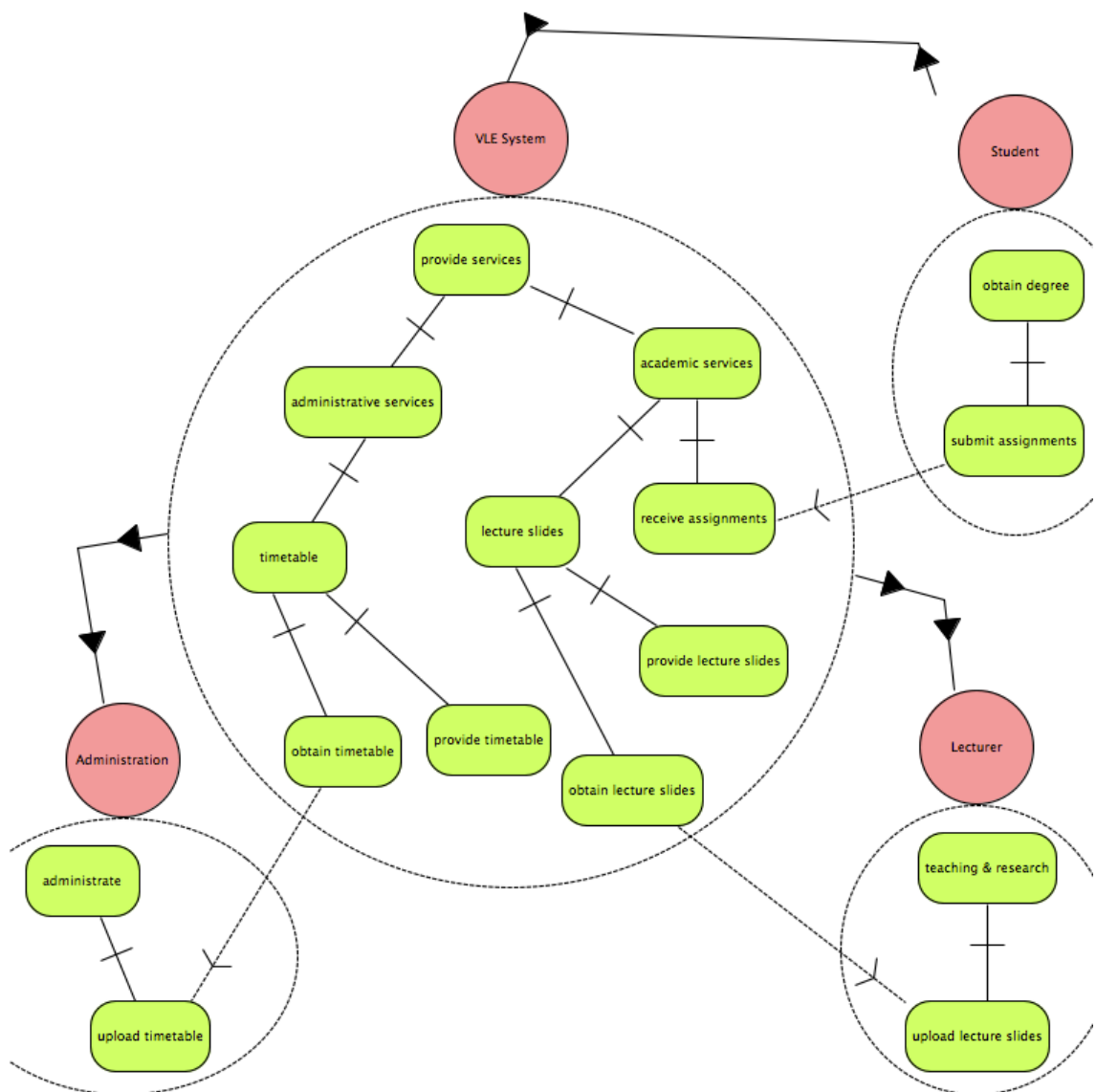


Figure 3.5: Dependency model example

riential trust, because there is previous direct experience with the administration that suggests that it will control the lecturer. The dependency on the university administration to upload the timetable is resolved with experiential trust as well, because there is previous direct experience with the administration that suggests that it will upload the timetable.

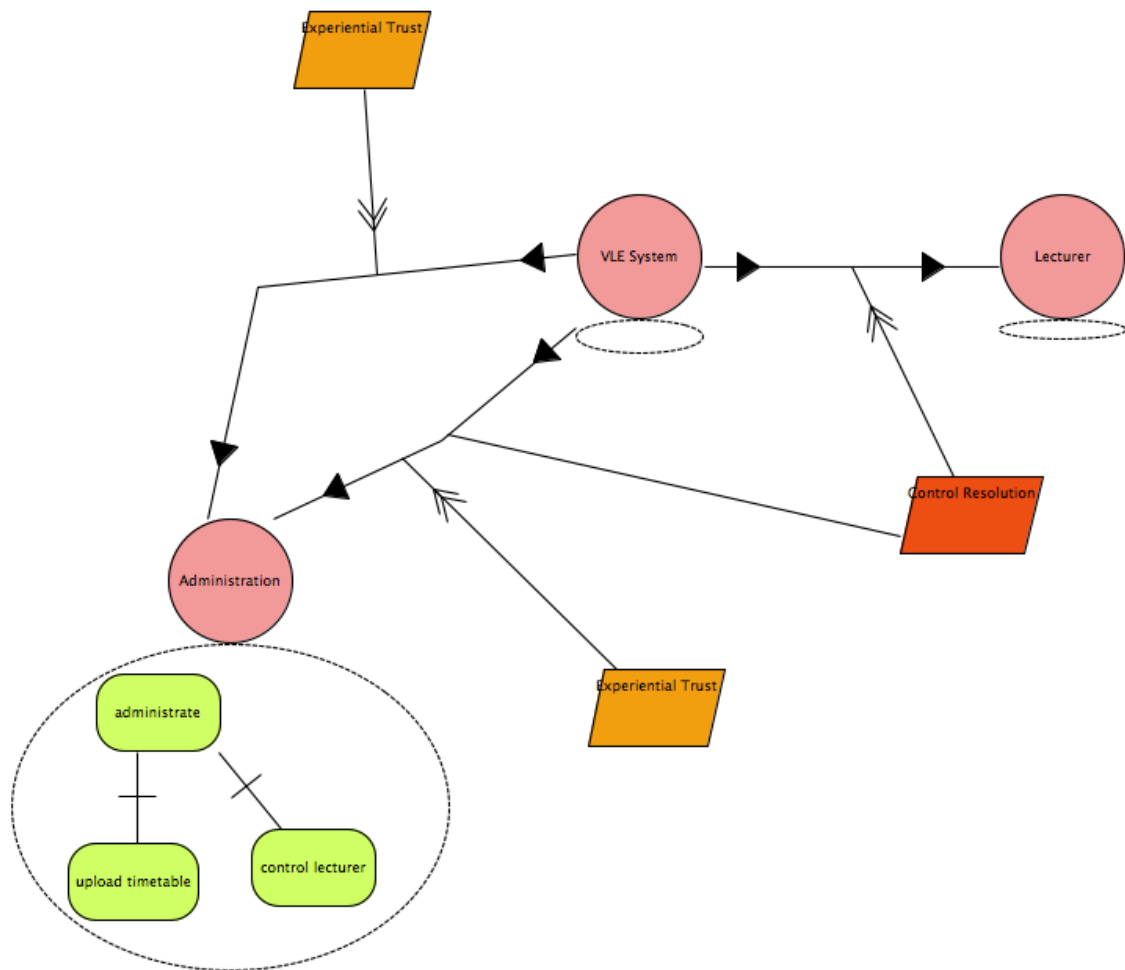


Figure 3.6: Resolution model example

Resolutions require entailments that are valid in order to have confidence in the fulfilment of the dependencies that will be reflected in the system trustworthiness. Each resolution has exactly one entailment, which can be true or false with respect to their validity. These concepts are illustrated in Figure 3.7 using the running example. The control resolution requires an entailment that the university administration is trusted to control the lecturer. The experiential trust resolution of the dependency introduced by the control resolution, requires an entailment that the developer trusts herself on whether the university administration will control the

lecturer or not. The other experiential trust resolution requires an entailment that the developer trusts herself on whether the developer will upload the timetable to the VLE system.

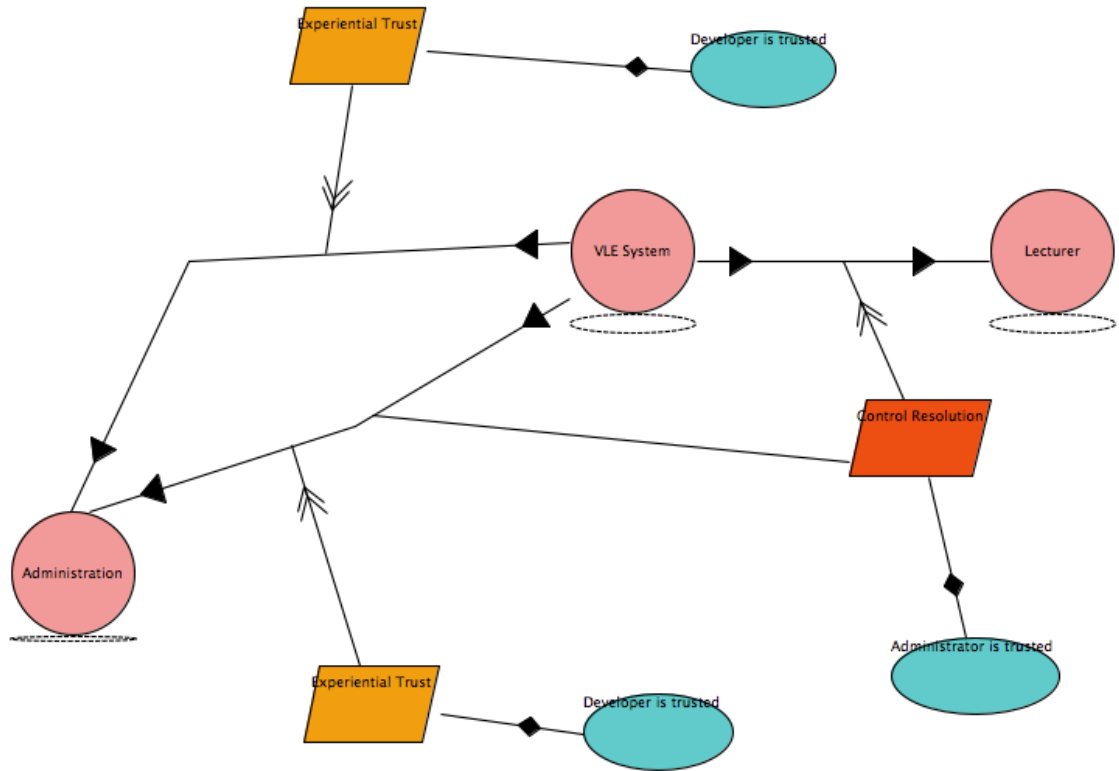


Figure 3.7: Entailment model example

If an entailment is not valid then the developer should investigate alternative solutions. As an alternative we have proposed in the previous section the identification and analysis of trustworthiness requirement. Dependencies that are not resolved by trust or control by a system environment actor can be resolved by control where the system-to-be is the controller. Such kind of control resolution will ensure the trustworthiness of the system-to-be. The analysis includes the identification of observation and deterrence capabilities for the system. As shown in the meta-model one system control resolution can have zero to many observations and zero to many deterrence. Of course, in order to be effective needs to have at least one observation capability and at least one deterrence capability. For each deterrence capability there are one to many goals of the dependee that are being prevented. Figure 3.8 illustrates how these concepts come together. The entailment that the developer trusts herself on whether the university administration will control the lecturer was not valid, and therefore a trustworthiness requirement is identified. The analysis of the trustworthiness requirement results in an additional system functionality to

check for every lecture if the lecture slides have been uploaded three days in advance as an observation capability. Also, the analysis results in an additional system functionality to prevent the lecturer from accessing his email account unless he has uploaded the lecture slides. The trustworthiness requirement acts as a threat and as a punishment in order to enforce the lecturer to behave in a specific way.

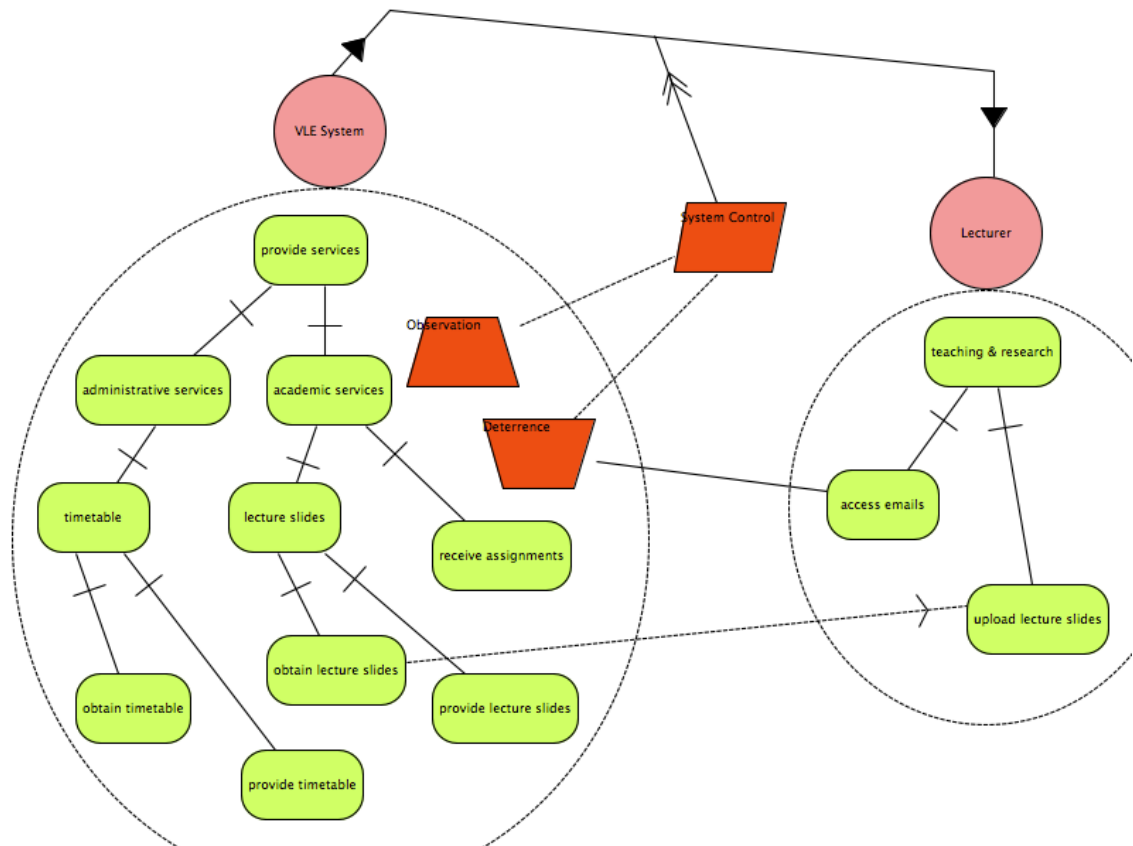


Figure 3.8: Trustworthiness requirement model example

### 3.8 Trustworthiness assessment model

Information system trustworthiness has two dimensions. The first dimension is if actors can achieve goals and the second dimension is if they will achieve them. A distinction has to be made between the technical system under development and the rest of the actors of the information system, which may include human or other technical actors.

Let us consider the case of the technical system under development. Its goals exist because they are needed by the system stakeholders. Some of these goals are accomplished by the system itself, but for others it needs to interact with other

system environment actors. To reason that the technical system-to-be can achieve its goals appropriate implementation mechanisms have to be modelled that show how the system will achieve the goals. There are multiple methods in the literature to deal with this task. On the other hand, for the goals that the technical system-to-be depends on other actors, first the capability of the actor achieving that goal has to be decided, again following available methods in the literature, and then to trust if the actor will actually achieve the goal once the system is put in operation. Back in the running example, the system trustworthiness of the VLE system in Figure 3.4 depends on the extent to which the high level goal, provide services, will be accomplished. As this high level goal is decomposed into smaller goals, the extent of accomplishment of the high level goal depends on the extent of accomplishment of the lower level goals, i.e., obtain timetable, provide timetable, obtain lecture slides, provide lecture slides, receive assignments. For the goals provide timetable and provide lecture slides, the developer can specify appropriate implementations and provide the system with the capability of achieving the respective goals. For the rest of the goals though a resolution of the dependency needs to be found. Consequently, any attempt to measure the trustworthiness of an information system needs to consider these two dimensions of trustworthiness.

In our proposed trustworthiness assessment model, any goal has a confidence level. If the goal is a lowest level goal then its confidence level  $C$  is:

- one("1") if the actor can achieve the goal by herself.
- zero("0") if the actor cannot achieve the goal by herself and there is no other actor to depend upon for this goal.
- the resolution level of the dependency if the actor cannot achieve the goal by herself and depends upon another actor for this goal.

The resolution level of a dependency, which shows the degree of confidence in the fulfilment of a dependency, is calculated as:

$$R = \left( \frac{ValidEntailments}{TotalEntailments} \right) \times ConfidenceLevel \quad (3.1)$$

ConfidenceLevel is the confidence level of the corresponding goal of the dependee actor.

Once the confidence levels of the lowest level goals are calculated then there is a bottom-up propagation of these values to the higher level goals until the highest level goals have a confidence level value. The propagation algorithm takes into account the decomposition share of subgoal to its parent goal.



ST is the system trustworthiness level, which shows how trustworthy the system is, i.e., how confident we are that the system will meet its requirements.

I is the importance of the high level goal to the overall system trustworthiness, and it is defined by the developer.

If n is a set of direct system dependencies then the system trustworthiness is calculated using the following formula:

$$ST = \left( \frac{\sum_{x=1}^n I_x \times C_x}{\sum_{x=1}^n I_x} \right) \times 100 \quad (3.2)$$

### 3.9 Chapter summary

This chapter has introduced the trust related concepts and a meta-model as the basis of a modelling language for trustworthy information systems development. During the development of systems the social architectures of trust need to be considered and reflected in the technical architectures of trust. Therefore, our goal is to capture trust relationships and to ensure that the design of the system conforms to them. As a result, the developer's trust assumptions during the analysis of the system need be identified, validated and if not validated then revised in order to lead to a system that is trustworthy for fulfilling its requirements. The proposed meta-model allows identifying both the direct and indirect trust relationships and reasoning about them in a structured and systematic way. Also, in cases where assumed trust relationships are not valid, control mechanisms are enforced in order to fill in the gaps in the chain of trust relationships.

## Chapter 4

# JTrust process

In this chapter, we describe a methodological process for modelling and reasoning about trust relationships, modelling and analysing trustworthiness requirements and assessing the system trustworthiness at a requirements level. The trust and control abstractions and the meta-model presented in Chapter 3 facilitate the development of trustworthy information systems by stipulating concepts that need to be taken into account. However, the meta-model alone is not adequate without the proper constructive techniques on how these concepts should be elicited. The JTrust process describes five activities as well as the artefacts produced. In particular, for each activity the steps and the relevant artefacts are specified. These activities are placed after the requirements elicitation and before the requirements specification. The Software and Systems Process Engineering Metamodel (SPEM) specification (*SPEM 2.0*) is used to specify the process, the activities, the steps, the artefacts and the roles involved. Figure reffig-process depicts an overview of the JTrust process with the following five activities:

- **Goal and dependency modelling:** to identify the stakeholders of the system-to-be along with their goals and their dependencies.
- **Resolution modelling:** to identify the resolutions of the dependencies along with possible introduced dependencies.
- **Entailment modelling:** to identify the trust assumptions that underlie the system under development and examine their validity.
- **Trustworthiness requirement analysis:** to identify the observation and deterrence functionalities that are required from the system in order to influence the fulfilment of dependencies.

- **System trustworthiness assessment:** to measure the system trustworthiness at a requirements stage.

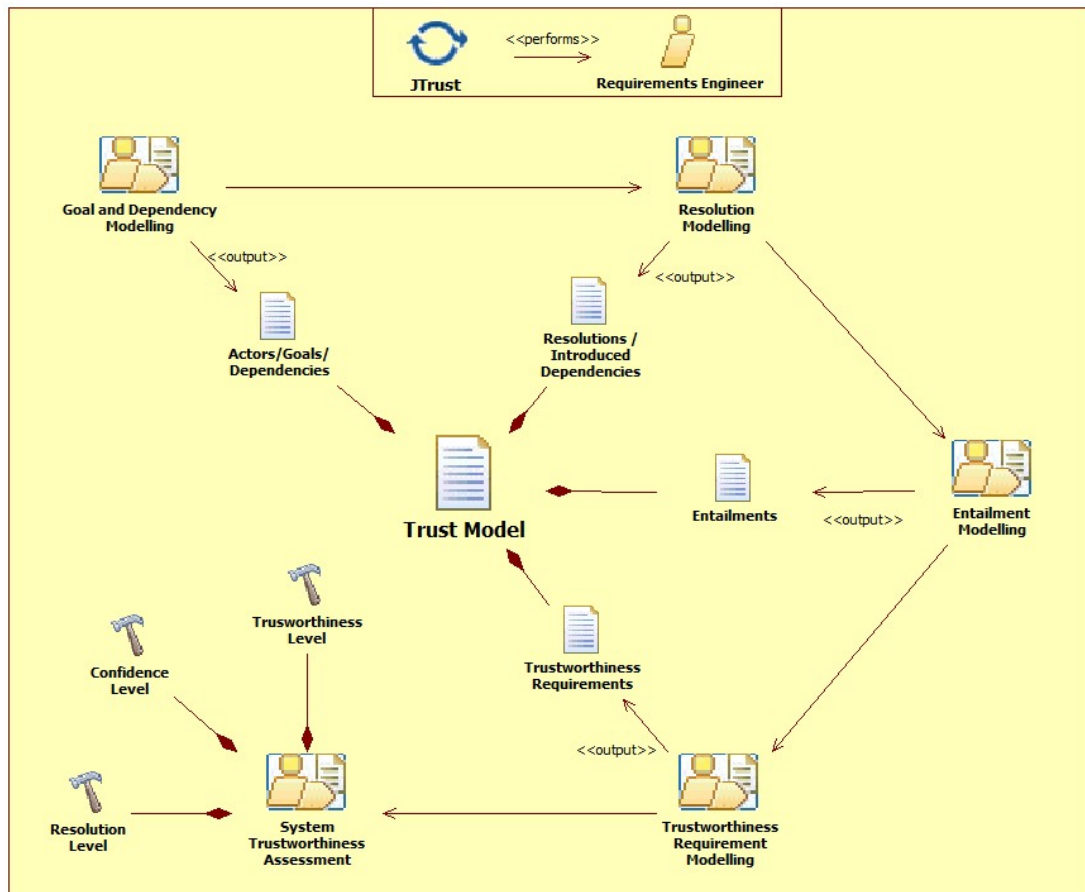


Figure 4.1: JTrust process

## 4.1 Activity 1: Goal and dependency modelling

The trust analysis of the information system begins with the modelling of actors' goals and dependencies. The main focus of this activity is to understand the organisational context and the system requirements. It contains three steps: actor identification; goal identification and refinement; and correspondences and dependencies identification. Figure 4.2 depicts the goal and dependency modelling activity with its steps and relevant artefacts.

In the first step, the actors of the information system need to be identified and modelled. This includes the technical system-to-be, human actors of the system environment and also other technical actors of the system environment. Human

actors are usually the stakeholders of the system, while the other technical actors are technical systems that the technical system under development is going to interact with. The developer can identify such information in organisational documents and interviews with the stakeholders. Every actor has a trustworthiness attribute, which influences the trustworthiness of the whole information system through the collaboration with other actors for the achievement of the information system goals.

In the second step the actors' goals are identified and refined in order to construct a goal structure for each actor. Goal identification is not an easy task. Sometimes they are explicitly stated by stakeholders or in preliminary material available to requirements engineers. Most often they are implicit so that goal elicitation has to be undertaken. Goals can also be identified systematically by searching for intentional keywords in the preliminary documents provided, interview transcripts, and so on (Van Lamsweerde, 2000). The preliminary analysis of current system, if it exists, is an important source for goal identification. Such analysis usually results in a list of problems and deficiencies that can be formulated precisely. Negating those formulations yields a first list of goals to be achieved by the system-to-be. Once a preliminary set of goals and requirements is obtained and validated with stakeholders, many other goals can be identified by refinement and by abstraction, just by asking how and why questions about the goals and requirements already available (Van Lamsweerde, 2000). The benefit of goal modelling is to support heuristic, qualitative or formal reasoning schemes during requirements engineering. Goals are generally modelled by intrinsic features such as their type and attributes, and by their links to other goals and to other elements of requirements model. In our proposed method goal refinement is done through AND/OR decompositions and every decomposition link has a share attribute. Also, every goal has a confidence level attribute. Additionally, the top level goals have an importance attribute. Therefore, once the goal model is constructed the developer has to specify the importance of the high level goals. Also, she needs to specify the share of every subgoal to his parentgoal. The share property denotes how much the parent goals is achieved with the achievement of the subgoal. For OR decompositions the share is "1". At the end the confidence levels of the lowest level goals need to be specified. The confidence level is "1" if the actor can achieve the goal by herself and "0" if she cannot achieve it by herself.

In the third step the correspondences and the dependencies need to be modelled. Goals cannot always be achieved by the actors themselves and such goals would have been specified with zero confidence level at the previous step by the developer. For these goals the developer has to seek for actors that can achieve them and specify the correspondence between the goal that cannot be achieved by the depender and the

corresponding goal of the dependee. Based on the correspondences between goals the dependencies are then specified. Therefore if the actor depends on another actor for a goal, then the confidence level is equal to the resolution level of the dependency.

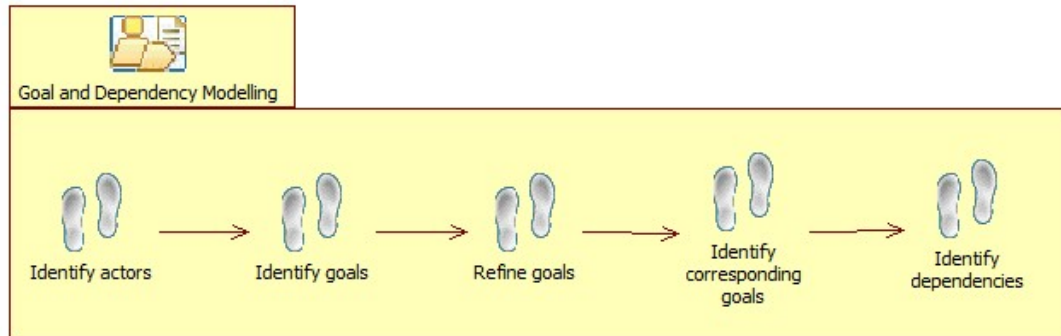


Figure 4.2: Goal and dependency modelling activity

```

1 Activity {kind: phase}: Goal and Dependency Modelling
2   ProcessPerformer {Kind: primary}
3     RoleUse: Requirements Analyst
4     WorkDefinitionParameter {Kind: in}
5       WorkProductUse: Organization Processes
6     Work DefinitionParameter {Kind: out}
7       WorkProductUse: List of Actors
8       WorkProductUse: Lists of High Level Goals
9       WorkProductUse: Goal Diagrams
10    Steps
11      Step: Study organization setting documents
12      Step: Identify actors
13      Step: Identify high level actor goals
14      Step: Model actors
15      Step: Model high level goals
16      Step: Refine actors' goals
17 ProcessPerformer {Kind: primary}
18   RoleUse: Requirements Analyst
19   WorkDefinitionParameter {Kind: in}
20     WorkProductUse: List of High Level Goals
21     WorkProductUse: Goal Model Diagrams
22   WorkDefinitionParameter {Kind: in}
23     WorkProductUse: List of dependencies
24     WorkProductUse: Actor Diagram
25     WorkProductUse: System Goal Diagram
26 Steps
27   Step: Model system and environment actors in the actor diagram
28   Step: Refine goals
29   Step: Identify goals that cannot achieved by their owners
30   Step: Identify correspondences among actors
31   Step: Model dependencies in the actor diagram

```

## 4.2 Activity 2: Resolution modelling

Once the dependencies among the actors of the information system have been modelled, the next activity is to identify and model the resolutions of the dependencies. This activity identifies the means of offsetting the uncertainty and potential vul-

nerabilities that dependencies introduce to the information system. The activity consists of two steps: model the resolutions and model the dependencies that the resolutions may introduce and resolve them as well until no further dependencies are introduced (Figure 4.3).

In the first step, each dependency is examined carefully in order to identify resolutions. To complete this task the developer has to consider the four types of trust resolution, i.e., experiential, reported, normative, and external, and the control resolution, and has to ask questions such as *"is there confidence in the fulfilment of the dependency because of experiential trust?"*, *"is there confidence in the fulfilment of the dependency because of reported trust?"*, *"is there confidence in the fulfilment of the dependency because of control?"*, and so on. Such questions will enable the developer to decide if there is confidence and what is the reason behind this confidence by modelling the appropriate trust and/or control resolutions. Then the respective resolutions are modelled in the trust model along with the dependencies that they resolve.

The next step of this activity is to model the dependences that reported trust and control resolutions are introducing. Reported trust resolution introduces a new dependency on the reporter, while control resolution creates a new dependency on the controller, if the controller is not the depender herself. Such dependencies need to be resolved in the same way following the first step of this activity. This is an iterative activity, which ends when there no dependencies left without resolution or if there are any unresolved dependencies then for these no resolution could be found. In the former case the developer can move on to the next activity to identify and validate the resolution entailments. In the latter case however, the trustworthiness requirement modelling activity should be followed.

```

1 Activity {kind: phase}: Resolution modelling
2   ProcessPerformer {Kind: primary}
3     RoleUse: Requirements Analyst
4   WorkDefinitionParameter {Kind: in}
5     WorkProductUse: List of Dependencies
6     WorkProductUse: Actor Diagram
7     WorkProductUse: System Goal Diagram
8   Work DefinitionParameter {Kind: out}
9     WorkProductUse: List of Resolutions
10    WorkProductUse: Resolutions diagram
11  Steps
12    Step: Identify outgoing system dependencies
13    Step: Identify type of resolution
14    Step: Model resolution
15    Step: Identify possible new outgoing system dependency
16    Step: Model possible new outgoing system dependency

```

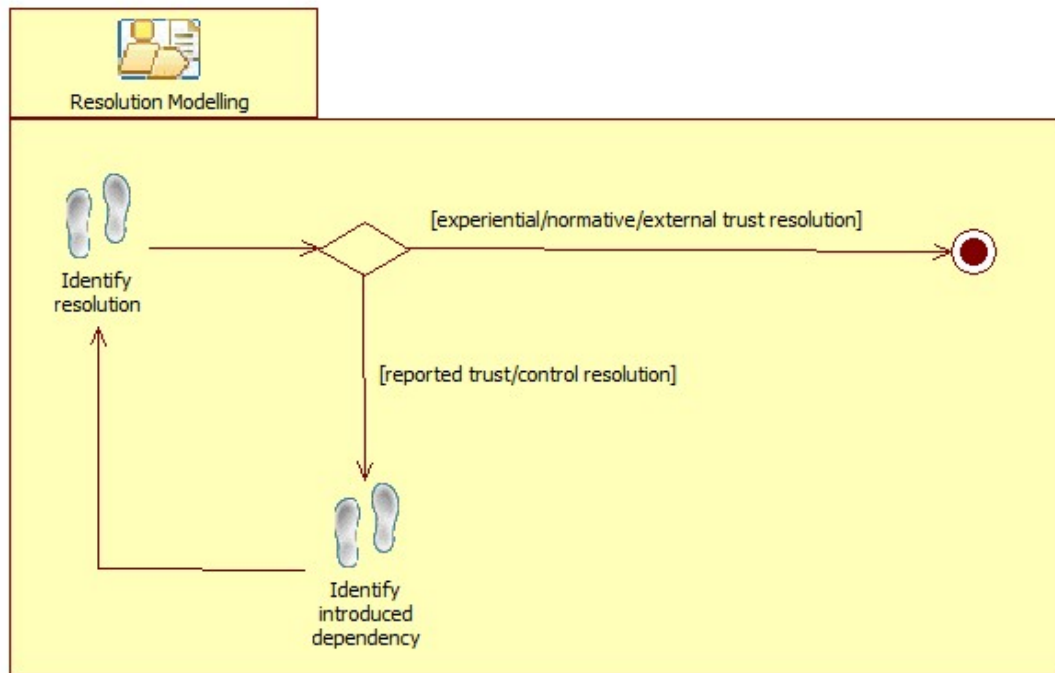


Figure 4.3: Resolution modelling activity

### 4.3 Activity 3: Entailment modelling

In the third activity the entailments are specified and validated. The entailments represent the trust assumptions that underlie the system under development. This activity contains two steps: the specification of the entailment according to specific guidelines, and the collection of evidence in order to examine the validity of the entailments (Figure 4.4).

In the first step of the activity the specification of the entailments is carried out based on the type of resolution. For each resolution identified in the previous activity an entailment is specified according to the following rules:

- Experiential trust requires an entailment that the trustor trusts himself.
- Reported trust requires an entailment that the reporter is trusted.
- Normative trust requires an entailment that the system environment norm is trusted.
- External trust requires an entailment that the external source of trust is trusted.
- Control based resolution requires an entailment that the controller is trusted.

In the second step the validity of the entailments has to be examined because if the entailments are not valid then there will be a potential vulnerability to the system that will affect its trustworthiness. The examination needs to be based on evidence. The developer can seek evidence in historical documents, carry out surveys and interviews. The entailment has a validity property that takes Boolean values true or false. If the entailment is valid then the validity property is set to true, otherwise it is set to false.

Based on the results of the entailments validation the resolution level of the dependency is calculated. Ideally it should be 100%, but it is up to the developer to decide what is the level upon which he considers that the dependency is resolved. If the dependency is resolved then there is confidence that the dependee will indeed behave as expected, otherwise there is no confidence that the dependee will behave as expected and additional functionality is required by the system to enforce the desired behaviour.

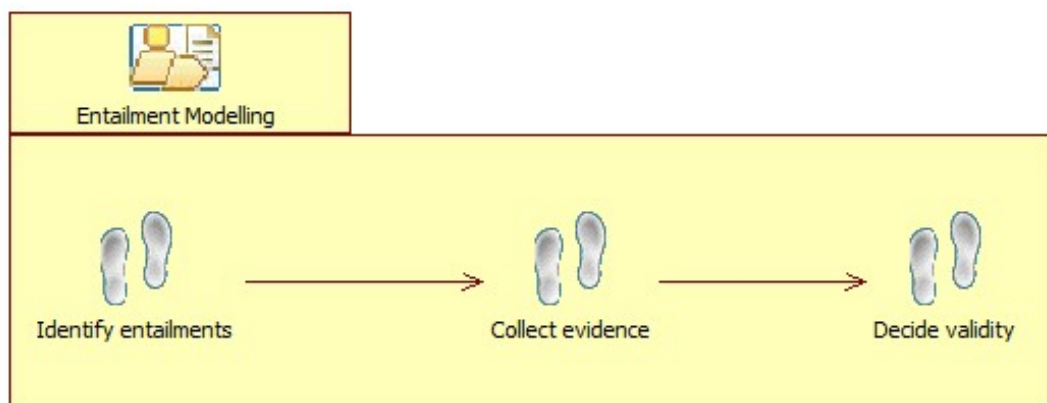


Figure 4.4: Entailment modelling activity

```

1 Activity {kind: phase}: Entailment Modelling
2   ProcessPerformer {Kind: primary}
3     RoleUse: Requirements Analyst
4   WorkDefinitionParameter {Kind: in}
5     WorkProductUse: List of Resolutions
6     WorkProductUse: Resolutions Diagram
7   Work DefinitionParameter {Kind: out}
8     WorkProductUse: Entailments diagram
9     WorkProductUse: List of valid entailments
10    WorkProductUse: List of invalid entailments
11 Steps
12   Step: Identify entailments based on resolutions
13   Step: Collect evidence regarding entailments
14   Step: Check validation of entailments
  
```



#### 4.4 Activity 4: Trustworthiness requirement analysis

The fourth activity of the JTrust process concerns the modelling and analysis of trustworthiness requirements. During the previous activity the validity of the entailments was examined. If the entailments are not valid the dependency is not actually resolved and alternative solutions need to be considered. An alternative solution is to add such functionality to the system-to-be that will act as a controller and enforce the fulfilment of an unresolved dependency. In essence, this is a control resolution of the dependency where the system-to-be is the controller. Therefore, this activity contains three steps: the identification of the control resolution, the specification of the observation functionality, and the specification of the deterrence functionality (Figure 4.5).

In the first step, every dependency is examined and if it has not been adequately resolved then a control resolution is modelled that resolves the dependency. In such a resolution the system under developer acts as the controller that will compel the dependee to behave as expected, i.e., fulfil a specific goal.

In the second step observation functionality for the system is modelled. This functionality enables the system to monitor if a specific actor is fulfilling a specific goal. To this end, a resource that enables the system to proceed to such a judgement has to be modelled as well. The developer in cooperation with the stakeholders identifies the relevant resource for that purpose.

The third and last step includes the modelling of the deterrence functionality for the system. This functionality will enable the system to compel actors to behave in a specific way. In particular the developer in cooperation with the stakeholders has to identify a goal of the dependee that should be denied if the dependee is not behaving in an expected way. The deterrence functionality along with the goal that is preventing are modelled.

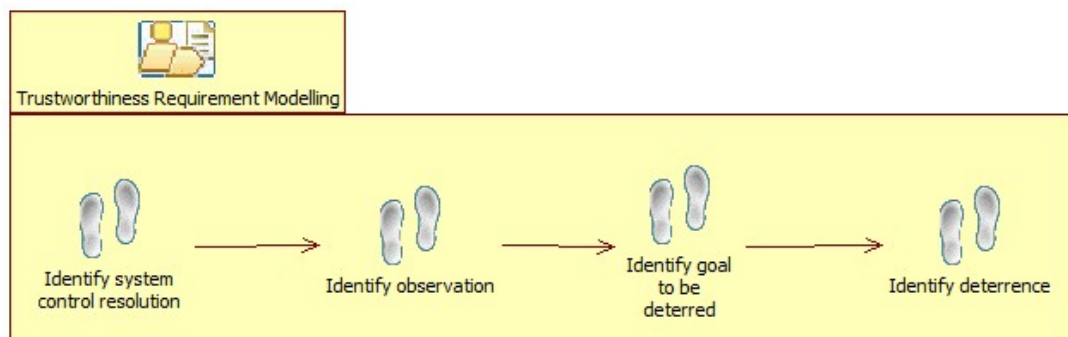


Figure 4.5: Trustworthiness requirement analysis activity

```

1 Activity {kind: phase}: Trustworthiness Requirements Modelling
2   ProcessPerformer {Kind: primary}
3     RoleUse: Requirements Analyst
4   WorkDefinitionParameter {Kind: in}
5     WorkProductUse: List of invalid entailments
6     WorkProductUse: Entailments diagram
7   Work DefinitionParameter {Kind: out}
8     WorkProductUse: List of trust requirements
9     WorkProductUse: Revised system goal diagram
10    WorkProductUse: Revised actor diagram
11    WorkProductUse: Revised actors' goal diagrams
12  Steps
13    Step: Trace back unresolved dependencies based on invalid entailments
14    Step: Assign control goal to system
15    Step: Decompose control goal to observation and deterrence goal

```

## 4.5 Activity 5: System trustworthiness assessment

This is the final activity of the JTrust Process, which measures the trustworthiness of the system-to-be. The measurement can be applied for all the actors of the information system but we believe the most useful approach is to measure the trustworthiness of the technical system-to-be. Assessing the system under development at the requirements stage is beneficial in order to identify potential vulnerabilities and address them as early as possible. Otherwise, any possible fix at late stage will cost more resources, such as time and money. Moreover, if potential vulnerabilities left unidentified and the system is put in operation then it may fail to meet its goals and users will not accept it. This activity contains four steps: assignment of importance level to the top level goals, assignment of confidence levels to the bottom level goals, calculation of the resolution levels, calculation of the system trustworthiness (Figure 4.6).

In the first step of this activity the top level goals of the system-to-be are assigned an importance value. The stakeholders are responsible for providing the importance values to the top level goals of the system-to-be. The range of importance is from zero to one.

During the second step the lowest level goals are assigned with confidence levels. The confidence level is one if the system can achieve the goal by itself, zero is the system cannot achieve the goal by itself, and if the system cannot achieve the goal by itself but depends on another actor to do so then the confidence level is the resolution level of the dependency. The confidence values will be propagated to the higher level goals until the top level goals.

The next step is the calculation of the resolution level of a dependency of the system-to-be with other actors. The resolution level is calculated according to the formula described in Section 3.8 (3.1).

The last step of this activity is the calculation of the technical system-to-be

trustworthiness level. It is calculated based on the formula described in Section 3.8 (3.2).

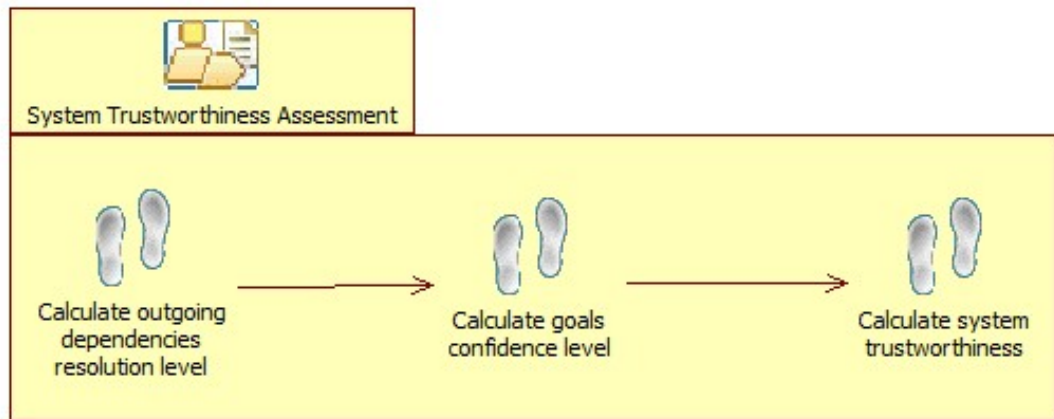


Figure 4.6: System trustworthiness assessment activity

## 4.6 Chapter summary

In this chapter, we have proposed a process for reasoning about trust relationships, analysing trustworthiness requirements and assessing the trustworthiness of the system-to-be. Section 4.1 presented the initial activity, which is focused on modelling the organisation context of the system-to-be and especially to capture the trust relationships as dependencies. In Section 4.2 the resolution modelling activity was described. It concerns with the identification of experiential, reported, normative, and external trust resolutions and control resolutions of dependencies in order to offset the uncertainty that the dependencies introduce. In addition, in this activity the dependencies introduced by reported and control resolutions are identified along with their resolutions. The next Section, 4.3 described the activity of entailment modelling. In particular, specific constructive techniques were described according to which an entailment is derived from each resolution. In addition, it is pointed out that the validity of the entailments needs to be examined by collecting relevant evidence. The trustworthiness requirement analysis was described in the following Section 4.4. The analysis consisted of modelling the trustworthiness requirement and specifying the observation and deterrence functionality required by the system-to-be in order to enforce the fulfilment of dependencies. Section 4.6 presented the formulas for measuring the trustworthiness of the technical system-to-be. The JTrust process is part of the requirements engineering phase of system develop-

ment. It can be applied along with the elicitation and analysis of other functional and non-functional requirements. It leads to the identification of trustworthiness requirements that will be added in the requirements specification of the system under development.

# Chapter 5

## JTrust tool

*If you cannot measure it, you cannot improve it*

---

Lord Kevin

In this chapter, we describe JTrust, a Computer Aided Software Engineering (CASE) tool to support developers in modelling and reasoning about trust relationships and analysing trustworthiness requirements. Also, it automatically assesses the trustworthiness of the system under development. It has a graphical user interface and the formulas for the system trustworthiness assessment are fully implemented. However, the tool is still a prototype and it can certainly be improved and optimised. First, we describe its architecture and the visual layout of the tool in Section 5.1, and then we describe the graphical notation of the JTrust concepts along with their properties in Section 5.2, followed by a description of the functionality of the tool in Section 5.3.

### 5.1 Tool architecture

For the implementation of the JTrust tool we used the Eclipse Integrated Environment (IDE) . Eclipse is a mature and extensible IDE framework that offers a powerful and flexible customisation framework. In particular, the implementation meta-model of our modelling language was defined as an Ecore model using the Eclipse Modelling Framework (EMF) . EMF is a modelling framework and code generation facility for building tools and other applications base on a structured meta-model. The meta-model is an independent artefact. Being defined using EMF means that it is decoupled from the code. Our implementation meta-model is shown in Figure 5.1.

In addition, the Eclipse Graphical Modelling Framework (GMF) was used to create our graphical editor. GMF builds on top of EMF and the Graphical Editing

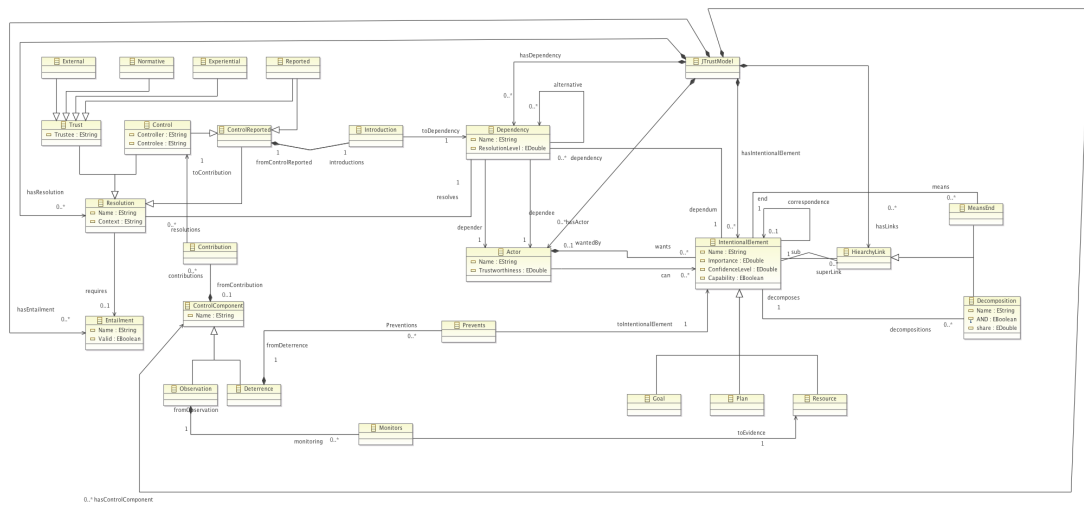


Figure 5.1: Implementation meta-model

Framework (GEF), to provide a generative component and runtime infrastructure. GMF build on the principles of Model-Driven Development (MDD), which advocate the use of design time models as the basis for code generation (Figure 5.2). It offers a layer of abstraction above the code level to define core properties of editors and the automatic generation of model editing code. This code is customised and expanded to support features specific to our proposed methodology. A visual layout of the JTrust tool is depicted in Figure 5.3.

The main graphical user interface (GUI) consists of the drawing canvas, the menu bar, the toolbar, and the properties panel. The drawing canvas is the area where the developer is incrementally constructing her JTrust model, while the menu bar has the standard functions, such as create new model, open model, save, and so on. The toolbar contains elements and links that correspond to concepts in the JTrust meta-model. The tool bar elements are laid out according to when these concepts might be used in a typical JTrust process.

The JTrust tool was developed in an iterative way. In the first phase, the tool was used only to model dependency resolutions in order to reflect the initial meta-model. During later stages, additional concepts, such as types of trust resolutions and observation and deterrence, were added to the meta-model and the tool was further developed to support these concepts. As the meta-model became more elaborate, additional functionality was incorporated into the tool. Further feedback

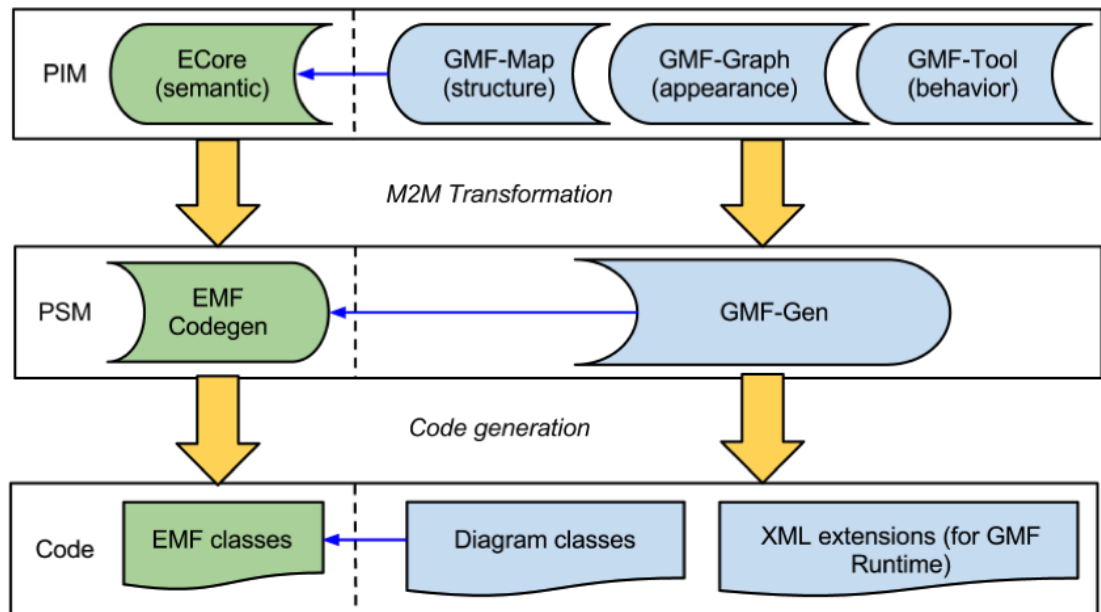


Figure 5.2: Model driven architecture

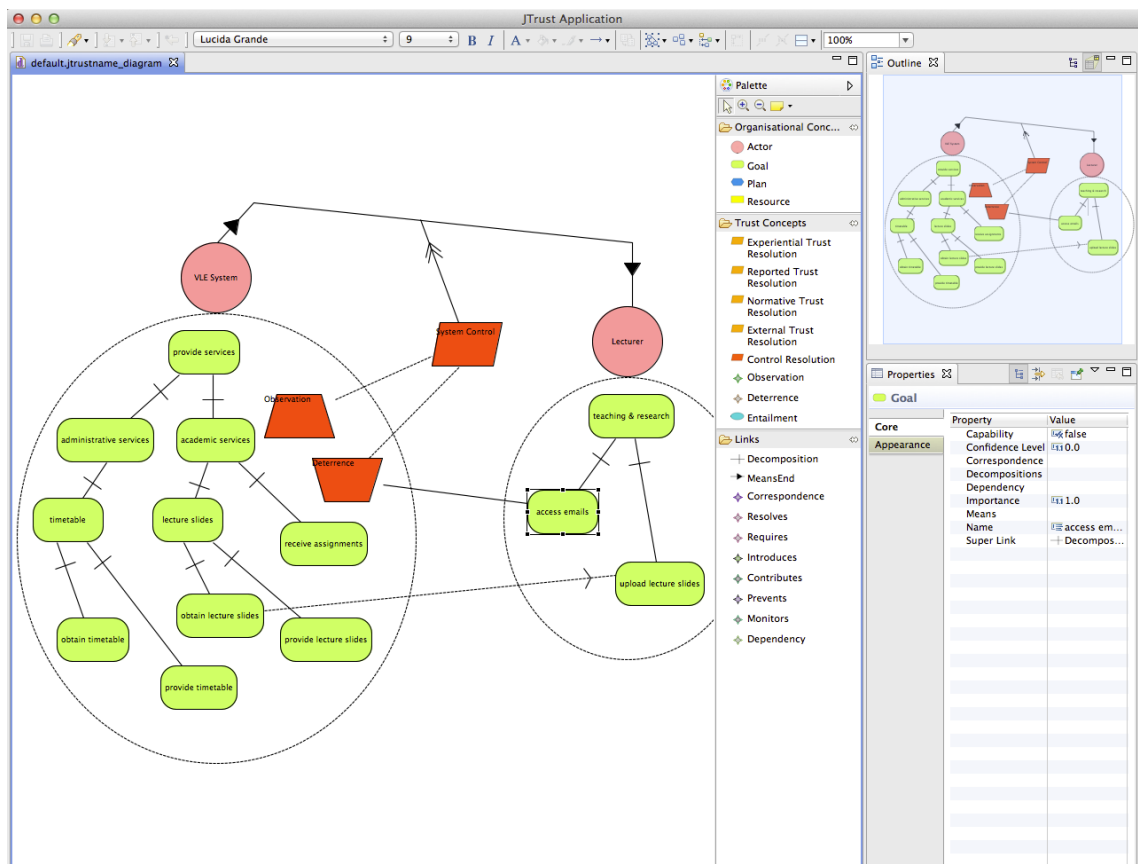


Figure 5.3: JTrust tool visual layout

about the methodology and the tool was obtained during the evaluation stage.

## 5.2 Concepts graphical notation

Given the difficulty associated with grasping new concepts and learning new notations, the tool and its artefacts need to be familiar. Adopting a tool should require no more cognitive overhead than learning how to use the techniques associated with JTrust methodology. When new notations are introduced, these need to be parsimonious in terms of visual complexity. Therefore, the new graphical notation was created in order to resemble the existing graphical notations of the actor and goal. The notations of the concepts used in the JTrust methodology are shown in Figure 5.4. In the next sections we present the properties of the main concepts of our proposed methodology.

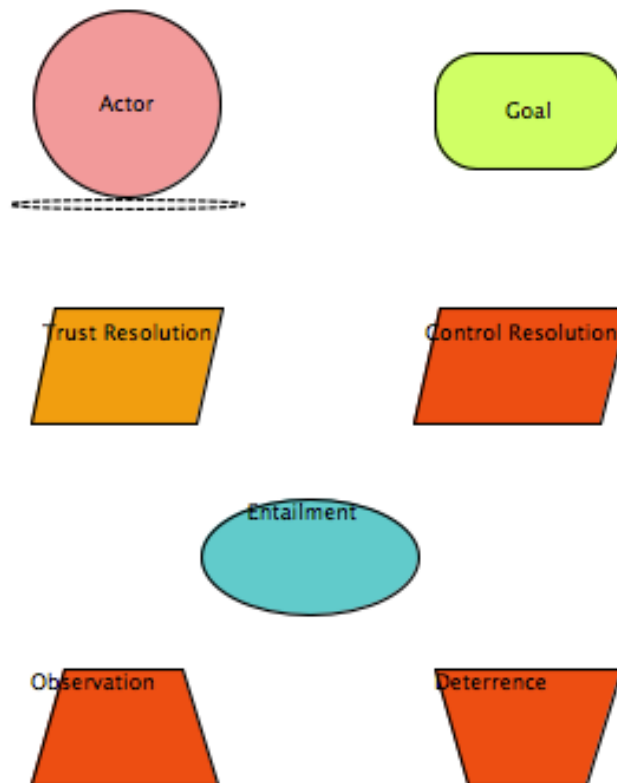


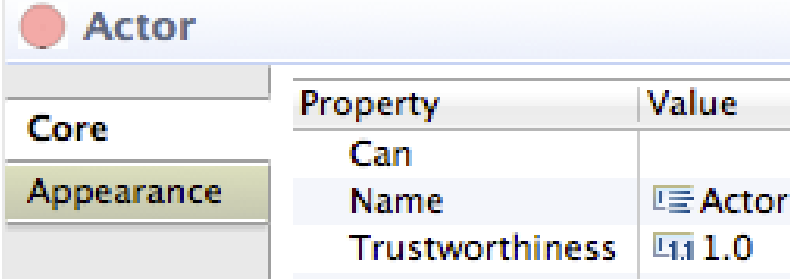
Figure 5.4: Concepts notation

### ACTOR

The actor has two properties (i) the name; (ii) and the trustworthiness (Figure reffig-actor). The name is the name of the actor specified by the developer, while the tool computes the trustworthiness of the actor automatically. Its value can range from



0% to 100%. The tool considers if the actor can achieve his goals and the resolution levels of the outgoing dependencies on other actors. If this actor represents the technical system under development then this property shows the trustworthiness of the system.



Property	Value
Can	
Name	Actor
Trustworthiness	1.0

Figure 5.5: Actor properties

### GOAL

The goal has a number of properties (Figure 5.6). The capability property denotes whether the actor can achieve the goal or not. The confidence level is automatically set to 1.0 if the actor can achieve it and to 0.0 if the actor cannot achieve it. In the latter case if there is also a dependency on another actor then the confidence level becomes the resolution level of the dependency. If the goal is decomposed then the confidence level is calculated based on the subgoals confidence levels and decomposition shares. Additionally, in the case of dependency the correspondence property shows the corresponding goal of the dependee actor. The decompositions property shows the decompositions of the goal, while the dependency property shows the dependency that the goal is part of. The importance property specifies how important is the goal for the system trustworthiness. The name property shows the name of the goal, while the super link property shows the decompositions that the goal is part of.

### DECOMPOSITION

The decomposition link has a property AND, which is Boolean, and denotes whether the decomposition is an AND or OR decomposition (Figure 5.7). The confidence contribution denotes how much confidence in the achievement of the parent the subgoal contributes. The value is automatically computed by the tool taking into account the decomposition share and the confidence level of the subgoal. The decomposes property shows the goal that is being decomposed, while the name property denotes the name of the decomposition. The share property is specified by the developer and represents the contribution of the subgoal in the achievement of the parent goal. Finally, the sub property shows the subgoal of the parent goal.

### TRUST RESOLUTION

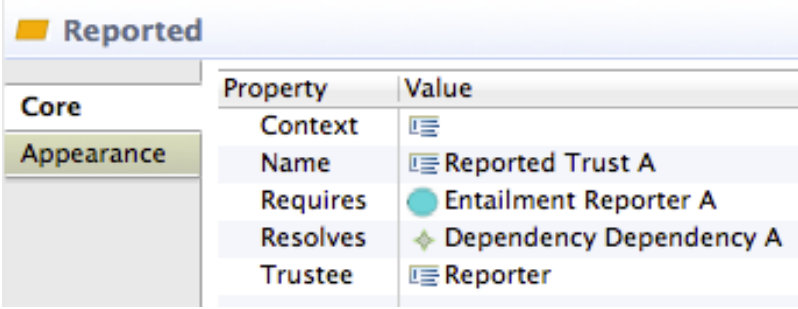
Goal		
Property	Value	
Capability	<input checked="" type="checkbox"/>	false
Confidence Level	<input type="checkbox"/>	0.5
Correspondence	<input checked="" type="checkbox"/>	Goal Goal B
Decompositions		
Dependency		
Importance	<input type="checkbox"/>	0.7
Means		
Name	<input type="checkbox"/>	Goal
Super Link	<input type="checkbox"/>	Decomposition to A

Figure 5.6: Goal Properties

+ Decomposition		
Property	Value	
AND	<input checked="" type="checkbox"/>	true
Confidence Contribution	<input type="checkbox"/>	0.2
Decomposes	<input checked="" type="checkbox"/>	Goal ParentGoal
Name	<input type="checkbox"/>	to A
Share	<input type="checkbox"/>	0.4
Sub	<input checked="" type="checkbox"/>	Goal Subgoal

Figure 5.7: Decomposition Properties

The trust resolution concept has a name property as usual that denotes the name of the trust resolution (Figure 5.8). Also, it has a requires property that shows which entailment is required from the resolution, while the resolves property shows which dependency is resolved. The last property trustee show who or what is trusted for the resolution.

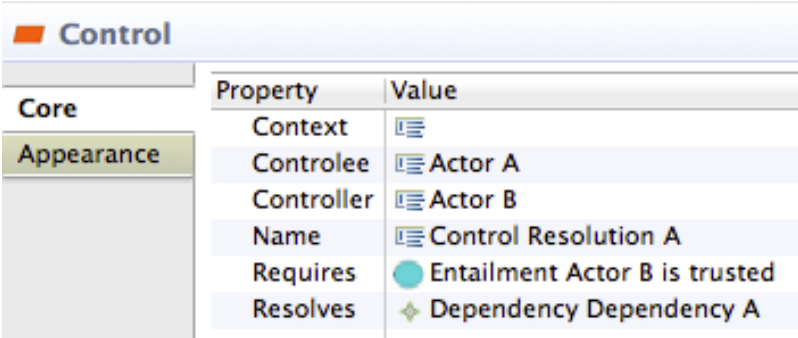


Reported	
Property	Value
Context	
Name	Reported Trust A
Requires	Entailment Reporter A
Resolves	Dependency Dependency A
Trustee	Reporter

Figure 5.8: Trust resolution Properties

## CONTROL RESOLUTION

The control resolution concept has a controlee and a controller property to denote the actor who is being under control and the controller actor respectively (Figure 5.9). Also, it has a name property to denote the name of the control resolution. The last two properties are the requires and resolves, which show which entailment is required by the control resolution and which dependency is resolved by the control resolution respectively.



Control	
Property	Value
Context	
Controlee	Actor A
Controller	Actor B
Name	Control Resolution A
Requires	Entailment Actor B is trusted
Resolves	Dependency Dependency A

Figure 5.9: Control Resolution Properties

## ENTAILMENT

The entailment concept has two properties, the name and the valid (Figure 5.10). The name property denotes the name of the entailment. The valid property is a Boolean property that is specified by the developer after she has examined its validity by collecting evidence and denotes whether the entailment is valid or not.

## DEPENDENCY LINK

Entailment	
Property	Value
Name	Reporter A is trusted
Valid	true

Figure 5.10: Entailment Properties

The concept of dependency has two properties *depender* and *dependee*, which show the *dependee* actor and the *depender* actor of the dependency respectively (Figure 5.11). Also, it has a *dependum* property that show goal of the dependency and a *name* property that denotes the name of the dependency. The resolution level is automatically computed by the tool taking into account the confidence level of the goal of the *dependee* and the validity of the resolution entailments of the dependency. The last property is the *resolutions* property, which shows the specified by the developer resolutions.

Dependency	
Property	Value
Alternative	
Dependee	Actor Actor
Depender	Actor Actor
Dependum	Goal Goal B
Name	Dependency A
Resolution Level	0.5
Resolutions	Reported Reported Trust A, Norm...

Figure 5.11: Dependency Properties

### 5.3 Trust tool functionality

JTrust tool is a tool for the construction and analysis of JTrust models as part of the trust analysis of the system under development. The main functionalities of the CASE tool are to support the modelling activities of the trust process. Therefore, the tool enables the developer to perform goal and dependency, resolution, entailment, and trustworthiness requirement modelling. The tool allows developers to draw JTrust graphical models using a palette of shapes. Standard features such as saving, zoom, cut, copy, and paste are provided as well. The tool also checks the syntactical correctness of a model during its development. For example, if the developer attempts to connect two concepts, that cannot be connected, with a link,

then the tool will prevent such action. In addition, the meta-model we presented contains OCL constraints. The meta-model provides the structure of the JTrust models, while the OCL constraints restrict the allowed syntax. These constraints enable to check if the model is well formed.

Furthermore, OCL was used to implement the formulas presented in this thesis.

1. Once the dependencies inside the socio-technical system have been specified and the entailments of the dependency resolutions have been defined as valid or invalid by the developer, then there is an algorithm that automatically calculates the resolution level of the dependency, from which the entailments originate. The OCL code that implements the formula 3.1 is as follows:

```

1  if self.dependum->isEmpty()
2  or self.resolutions->isEmpty()
3  or self.resolutions->collect(requires)->isEmpty()
4  or dependum.ConfidenceLevel=0.0
5  then 0.0
6  else resolutions->collect(requires)->collect(Valid)->
7      count(true) / resolutions->size() * dependum.ConfidenceLevel
8  endif

```

2. Once the confidence levels of the lowest level goals of an actor have specified by the actor or automatically by the tool in case of existence of dependencies during the previous calculation, then the tool automatically computes the confidence levels of the parent goals until the top level goals. In other words there is a bottom-up propagation of the confidence levels in the goal structure. The OCL code that implements the formula is as follows:

```

1  if self.decompositions->notEmpty()
2  then if decompositions->collect(AND)->count(false) = 0      then
3      decompositions->collect(ConfidenceContribution)->sum()
4      else self.decompositions->
5          sortedBy(ConfidenceContribution)->last().
6          ConfidenceContribution
7      endif
8  else if self.Capability
9      then 1.0
10     else if self.correspondence->isEmpty()
11         then 0.0
12     else if self.correspondence.dependency.dependum->
13         isEmpty()
14         or self.correspondence.
15         dependency.resolutions->
16         isEmpty()
17         or self.correspondence.
18         dependency.resolutions->
19         collect(requires)->isEmpty()
20         or self.correspondence.
21         ConfidenceLevel=0.0
22     then 0.0
23     else self.correspondence.dependency
24         .resolutions->
25         collect(requires)->collect(
26             Valid)->count(true) /
27         self.correspondence.dependency.
28         resolutions->size() *
29         self.correspondence.
30         ConfidenceLevel

```

```

22         endif
23     endif
24 endif
25

```

3. Once the importance levels of the top level goals of an actor have been specified by the developer, the tool automatically computes the trustworthiness of the actor. If the actor is the system under development then the trustworthiness value is the trustworthiness of the system-to-be. The OCL code that implements the formula 3.2 is as follows:

```

1  if wants->notEmpty()
2      then
3          if self.wants->select(superLink->isEmpty())->
4              collect(Importance)->sum() > 0
5              then (self.wants->select(superLink->isEmpty())->
6                  collect(ConfidenceLevel*Importance)->sum()/self.
7                      wants->
8                      select(superLink->isEmpty())->collect(Importance)->
9                          sum()*100
10                 else 1.0
11             endif
12         else 1.0
13     endif

```

## 5.4 Chapter summary

In this chapter the JTrust CASE tool that assists the developer in using the JTrust methodology was presented. We have described the tool development and the visual layout of the tool, the graphical notation of the JTrust concepts and their properties, and the functionality of the tool that enables the developer to construct all the required models of the methodology and provides automatic computation of the trustworthiness level of the system under development.

## Part III

# Evaluation and Conclusions

## Chapter 6

# Evaluation

This chapter focuses on the evaluation of the proposed JTrust methodology, in particular, assessing the strengths and the weaknesses and demonstrating the advantages. This part of the research is empirical research that includes research questions that are focused on the way that the world is and are the last three research questions of this research:

- RQ4: How well does the methodology support modelling and reasoning about trust relationships?
- RQ5: How well does the methodology support trustworthiness requirement modelling and analysis?
- RQ6: How well does the methodology assess the system trustworthiness at a requirements level?

Therefore this study aims to provide strong support for the validity of the methodology. We adopted two methods of evaluation for the methodology of this thesis that are depicted in figure 6.1. The first evaluation method was a confirmatory qualitative case study to test the developed theory, i.e. the JTrust methodology. Confirmatory because a case study can be used for both generating and testing hypotheses (Flyvbjerg, 2006; Seaman, 1999) and the aim was to build a convincing body of evidence to support the propositions of this thesis. And qualitative because the assessment was based on the required features that the JTrust methodology provided. Furthermore, it included observational and historical data collection techniques, because relevant data were collected as the project developed and also from projects that had already been completed.

The case study that was selected was the e-health care services of the National Health Service (NHS) in England (NHS, 2013a). The NHS e-health services are



centred on the Summary Care Record (SCR), which envisages improving the safety and quality of patient care. As of 26/04/2013 almost 27 million records had been created across England. Having a SCR gives authorised healthcare staff a quicker way to get important information about the patient, in an emergency or out-of-hours situation. The SCR is an electronic record which contains information about the medicines the patient takes, allergies the patient suffers from, and any bad reactions to medicine the patient had. The case study is appealing for a number of reasons; they are real, safety critical concerns and involving important trustworthiness requirements both in terms of security and privacy. Information might be concerning an abortion, a psychiatric care, or sexual transmitted diseases, which in case of disclosure can cause social embarrassment, prevent us from getting a job or influence our medical insurance. Therefore, the case is an appropriate one as it is a typical and critical case for testing a well-formulated theory because if theory holds for this case, it is likely to be true for many others. In terms of the validity of the study, data and method triangulation techniques were used to mitigate any threats and feel confident in the evaluation.

The evaluation method were chosen according to the selection criteria identified by the DESMET project (Kitchenham, Linkman, and Law, 1997). So, the confirmatory case study was selected because the benefits could be observed on a single project. Moreover, software engineering is a multidisciplinary field involving areas where case studies normally are conducted, such as psychology, sociology, and political science. This means that many research objectives in software engineering research are suitable for case study research (Runeson and Höst, 2009), and that there was the need to investigate not only methods and tools but also the social and cognitive processes surrounding them (Easterbrook et al., 2008). In these areas the objectives are to increase knowledge about personal, social, and political phenomena in their context, which is similar with our objectives to increase knowledge regarding the practitioner's ability, knowledge, understanding, to capture trust relationships during information systems development. Case study research is appropriate for situations where the context is expected to play a role in the phenomena.

The second evaluation method was according to Zelkowitz and Wallace (1998) a quantitative survey. The subjects were academics, researchers, and postgraduate students that used the methodology and the tool and were asked to provide information about the methodology and the tool in order to investigate the quantitative impact. It included controlled data collection methods, in particular a questionnaire based survey, that provided multiple instances of an observation for statistical validity of the results. The collected information was then analysed using standard statistical techniques. The analysis focuses on the methodology's perceived useful-

ness and ease of use. The results will be evaluated by a combination of quantitative and qualitative analysis. The survey was also used for qualitative evaluation, which was a feature-based evaluation.

Also, the quantitative and qualitative survey methods were selected because there was availability of participants with experience of using methodologies and tools and also participants were prepared to learn about new methodologies and tools. They can confirm that an effect generalises to many projects. Also, this is a more satisfactory means of evaluation of the benefits than a simple review of the methodology by the participants. It is also very useful for identifying practical problems with the methodology, such as ambiguities or missing conditions to name a few.

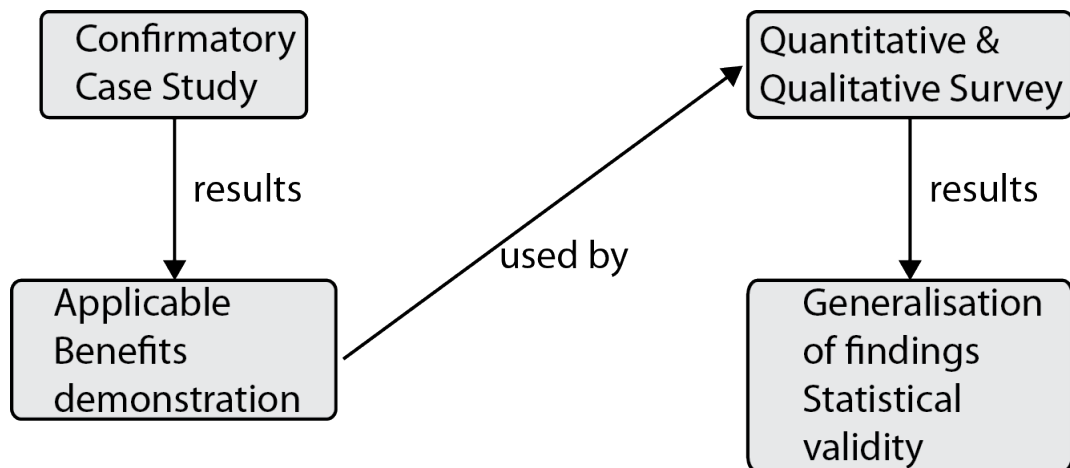


Figure 6.1: Evaluation approach

## 6.1 Study 1: Evaluation of JTrust by case study research in the health care domain

This chapter reports the experience of conducting a case study on the application of JTrust methodology in the health care domain in the UK. The goals of this chapter are to evaluate the JTrust methodology and to show that it efficiently enables practitioners to model and reason about trust relationships and assess the trustworthiness of the system under development at a requirements stage. We wanted to demonstrate JTrust in action, so our preferred evaluation method was a confirmatory qualitative case study. A project was selected and monitored, and data was collected over time. It is used to demonstrate the applicability and the advantages of the methodology and serves as a first test before a more formal validation.

Therefore, this chapter illustrates and assesses the various techniques described in the preceding chapters on a real case study of a significant size and complexity.

According to Zelkowitz and Wallace (1998) taxonomy for validating new methodologies, this study is using historical data collection methods and as a result it can also be characterised as literature search based method of validation, as there was also collection of data from projects that have already been completed and analysis of papers and other documents that are publicly available. In particular, some conclusions were drawn based upon surveys available in the literature and documents related to the selected case study.

In addition, based on the Zelkowitz and Wallace (1998) taxonomy the study can also be characterised as an assertion type of case study because the researcher was both the experimenter and the subject of the study. This was the case because of time and cost constraints. But, it served as a first step towards a future more formal evaluation.

The process steps that our case study included were the following (Verner et al., 2009; Runeson and Höst, 2009; Kitchenham et al., 2002; Wohlin et al., 2012):

- Research initiation. This phase included the defining of the research objectives, i.e. what to achieve, performing a deep literature review, and decision on the appropriateness of the case study. The appropriateness was decided based on the findings from the research objectives and the literature review.
- Case study administration, which dealt with legal, publishing, and scheduling issues. It was an "ad hoc" phase that was done in parallel with all the other phases. It included the review of legal agreement, the identification of publishing criteria, and dealing with partners and their schedules.
- Case study focus, which included activities such as the identification of the boundaries of the case study. Also, the identification and selection of feasible cases. The case was expected to be "typical", "critical", and "revelatory". Also, what documents should be accessed were selected.
- Design case study plan, which includes ensuring strategy for data validity, the minimisation of the confounding factors, defining the data collection strategy, process and methods, procedure and protocols, how the results will be analysed and designing the case study plan step by step.
- Data collection, which basically included observational methods where data was collected during the development of the project. Furthermore, it included the collection of data from multiple sources, such as literature search to examine previously published studies, legacy data to examine to examine data from

completed projects, static analysis to examine the structure of the developed product, and lessons learned that examined qualitative data from completed projects.

- **Data analysis.** This phase included the evaluation and the conclusion with respect to the case study. However, as case study methodology is a flexible design strategy, there is a significant amount of iteration over the steps, and the data collection and analysis were conducted incrementally. During the whole process there was constant updating of research notes, and the structure of the case study report was in the form of a single case study narrative report.

### 6.1.1 Case study design

#### CASE STUDY OBJECTIVES

The objectives of the case study were defined in a way to provide answers to the research questions of the evaluation research and are confirmatory and/or improving. Therefore the objectives of the case study were to:

- O1:** Apply the JTrust methodology in the health care domain.
- O2:** Evaluate the applicability of JTrust methodology for modelling and reasoning of trust relationships and requirements in health care domain.
- O3:** Evaluate the applicability of JTrust methodology for assessing the trustworthiness of the system under development at a requirements stage.
- O4:** Evaluate the efficiency of JTrust methodology.
- O5:** Improve our understanding about the issues relating to the JTrust methodology with the intention of improving it.

#### CASE SELECTION/CONTEXT - E-HEALTH CARE SERVICE IN ENGLAND

The case study that was selected was the e-health care services of the National Health Service (NHS) in England (NHS, 2013a). The project was launched in 2002 and its aim was "to reform the way that the NHS in England uses information, and hence to improve services and quality of patient care" (Office, 2011). Central to the programme was the creation of fully integrated electronic care records system that is designed to reduce reliance on paper files, make accurate patient records available at all times, and enable the transmission of information between different parts of the NHS. The system was intended to comprise a health care record for each NHS patient that will contain full details of the patient's medical record and treatment,

that will be accessible to a patient's GP and local community and hospital care settings. Such health care systems will cost less and will be more effective with the use of healthcare networks (Bangemann et al., 1994). Also, health care services can significantly be enhanced with the use of the Internet. The program envisages providing such advance services to patients and the NHS employees and ultimately to improve the quality of patient care. As the delivery of patient care is now often shared across a number of NHS clinical or business areas and suppliers, so the effective linking up and flow of information related to a patient has become even more important.

The NHS e-health services are centred on the Summary Care Record (SCR). The SCR envisages improving the safety and quality of patient care. It will give healthcare staff faster, easier access to reliable information about the patient to help with his treatment. The patient information was shared by letter, e-mail, fax, or phone. At times, this could be slow and sometimes things got lost on the way. As of 26/04/2013 almost 27 million records had been created across England. The SCR is held nationally, while the Detailed Care Record is held locally at places that treat the patient regularly, like the GP or the local hospital. Having a SCR gives authorised healthcare staff a quicker way to get important information about the patient, in an emergency or out-of-hours situation. The SCR is an electronic record which contains information about the medicines the patient takes, allergies the patient suffers from, and any bad reactions to medicine the patient had. This information could make a difference to how a doctor decides to care for you, for example which medicines they choose to prescribe for you. It is very helpful for pharmacists as well to understand the core essential information about the patient. It is optional for patients to add extra information to their SCR, such as current illness or care plan. Only health care staff involved in the patient's care can see the patient's SCR. Healthcare staff will only see the information they need to do their job, and they will ask your permission every time they need to look at the patient's SCR. If they cannot ask the patient, for example if he is unconscious, they may look at the patient's SCR without his permission. If they do this, they will make a note on the patient's record to say why they have done this.

By law, everyone in, or on behalf of, the NHS must respect the patient's privacy and keep all the patient's information safe (NHS, 2011b). If someone accesses the patient's record unexpectedly, for example if someone who does not usually treat the patient looks at his record, an alert will be sent to a member of staff. They will investigate and, if it is found that this was unreasonable, they will let you know. To keep the patients information secure and confidential the NHS employs several measures. Any member of staff being given access to national systems, which

hold health information, has a smartcard along with a username and password. Besides smartcards other measures used are the recording permission to access, access control, and audit trails.

For the delivery of care record systems the department of health awarded five 10-year contracts to four suppliers: BT in London; Accenture in the East and in the North East; Computer Sciences Corporation in the North West, and West Midlands; and Fujitsu in the South. The aim was for care records systems to be delivered to all NHS trusts and GP practices. Currently, the program, which was called "National Program for IT" has been dismantled into its separate components parts, and the revised completion date is 2018 because of difficulties in implementation, ethical issues that cause delays, and contractual issues. However, the vision remains for a paperless NHS in UK, under the supervision of a newly set up organisation called "Health and Social Care Information Centre", but with a balance between standardisation across the NHS and the desire for local ownership and flexibility (Public Accounts, 2013).

The main purpose of the e-health care services is to make the patients' data accessible from anywhere. For this purpose, the patients' data is going to be stored nationally. Employees of the NHS, such as GPs, can have access to the patients' data in order to provide effective health care. After the patient has been examined the GP must update the Summary Care Record (SCR), which contains the patients' data, and electronically sign the prescription that will enable the patient to collect the medicine from a pharmacy (Figure 6.2). The GP is allowed to cancel the prescription at any time until it is dispensed, but when doing so he must inform the patient for the reason of the cancellation. In addition, the pharmacy needs to electronically confirm what has been dispensed and to electronically submit endorsement messages to the reimbursement agency for dispensed prescriptions to support the reimbursement process.

The case study concerns the patients' health care record in the health care domain, and specifically the former National Program for Information Technology (NPfIT) in United Kingdom. The case study is appealing for a number of reasons; they are real, safety critical concerns and involving important trustworthiness requirements both in terms of security and privacy. Information might be concerning an abortion, a psychiatric care, or sexual transmitted diseases, which in case of disclosure can cause social embarrassment, prevent us from getting a job or influence our medical insurance. Moreover, there is a lot of history medical privacy issues such as drug stores providing medical information to marketing companies, employers taking employment decisions based on medical information, employees or individuals selling medical information (Rohm and Milne, 2004). This is not the

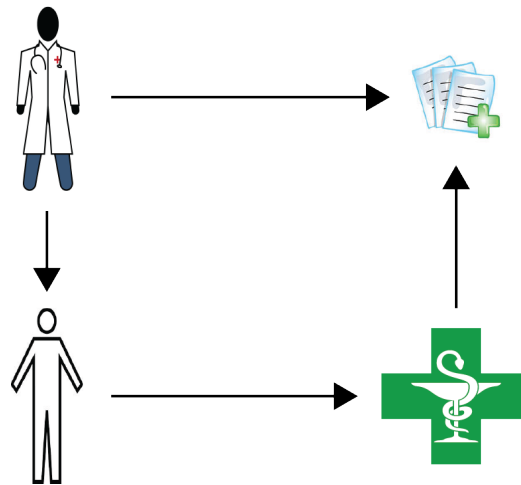


Figure 6.2: e-Health scenario

case only in Britain, but also in the U.S.A. where the majority of the patients are concerned with issues varying from use of their medical information for marketing purposes to identity theft or fraud (McGraw et al., 2009). Therefore, privacy for the programme is very important and we have to provide assurance for privacy that will ensure trustworthiness and enable the development of trust by the users. Therefore, the case is an appropriate one as it is a typical and critical case for testing a well-formulated theory because if theory holds for this case, it is likely to be true for many others.

#### UNIT OF ANALYSIS - A TYPICAL SCENARIO

To assess the effectiveness of the trust process we apply the trust based concepts and the process in the case of e-health care services of the National Health Service (NHS) in England. Then the trust relationships will be identified and examined if the trust assumptions are justified. If not, then additional functionality will be proposed to make the system trustworthy. Also, towards the end, there was an assessment of the system trustworthiness with the intention the system to be as trustworthy as possible.

The technical system under development for the NHS will be interacting with other components of that whole information system, either human or technical. These interactions constitute dependencies between the system and the other components and vice versa. The unit of analysis is the dependency between the technical system and other components of the information system. As there were multiple dependencies there were multiple instances of this unit of analysis. Another unit of analysis will be the identification of trustworthiness requirements. Similarly, there were multiple instances of this unit of analysis. The last unit of analysis was the

assessment of trustworthiness of the system under development.

### DATA COLLECTION METHODS

Several methods of data collection were used in the case study (Figure 6.3). The developed diagrams using the JTrust methodology were collected, along with qualitative notes that were taken on those diagrams. Also, documents related with the existing development of the NHS system were collected. Also, information was gathered by discussing with staff from BT. Finally, documents related to past surveys were collected.

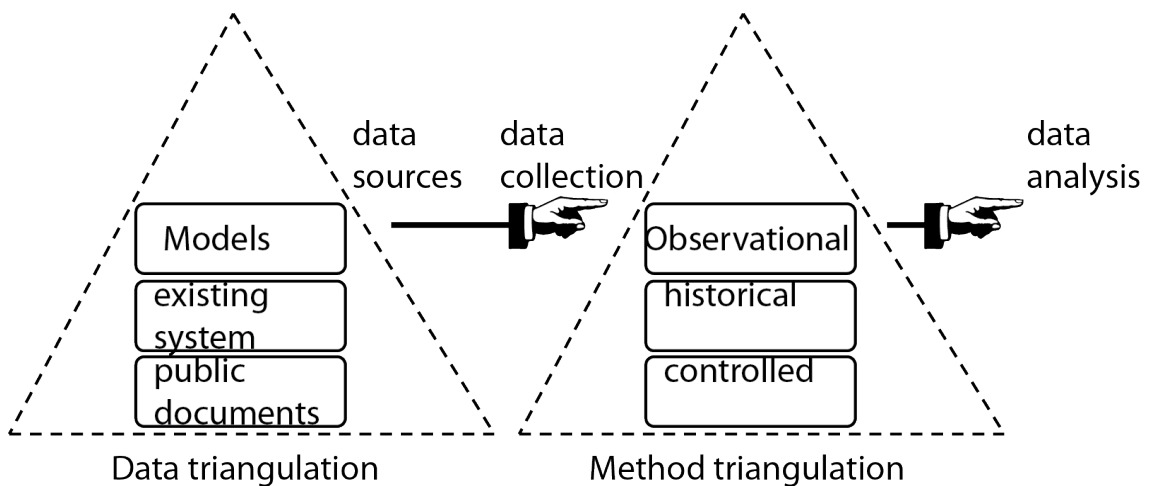


Figure 6.3: Case study collection methods

### DATA ANALYSIS METHODS

The existing functionalities of the NHS system were compared with the identified functionalities using the JTrust methodology. Then, there was an evaluation of the non-quantifiable benefits of the JTrust methodology by gathering qualitative information from the comparison.

### CASE STUDY VALIDITY

The validity of the study is discussed in this section, since it could not be finally evaluated until the analysis step. There are four aspects of validity (Runeson and Höst, 2009; Wohlin et al., 2012; Easterbrook et al., 2008).

Construct validity was focused on whether the operational measures for the concepts being studied were correct. Therefore, to improve construct validity the operational measures with regards to the confidence in the fulfilment of a system dependency were developed by using multiple sources of evidence, such as literature and expert opinion.

Our case study was by nature comparative. We tried to reduce the expectation bias on the case study results by identifying a valid basis for assessing the results.



To this end, we compared the actual implementation of the NHS e-health system developed with another method and the potential implementation of the NHS e-health system developed with our proposed method. Therefore, we improved the internal validity as we compared our proposed method while using the same information system. Furthermore, the comparison was on the basis of identifying the system dependencies and contrasting whether there has been any reasoning about their fulfilment.

External validity focuses on whether our claims for the generality of the results are justified. To ensure external validity we used data triangulation to limit the effects of one interpretation of one single data source, data were collected from multiple sources. So, in terms of gathering information about the NHS System various data were collected from documents from BT, NHS England, UK Department of Health, and research publications about the NHS e-health system. Based on this data the JTrust models were constructed and used as a data source. This way the conclusions reached are stronger than conclusions based on a single source. Data triangulation ensured the external validity of our case study, which justified the generalisation of our results.

Another validity concern is the representativeness of the selected case study, which ensured the external validity of our case study. An e-health information system was selected, as it is a typical information system, which its trustworthiness depends a lot on other interacting entities such as a doctor. Also, its trustworthiness is of paramount importance because it has serious consequences on citizens' health. Furthermore, when we chose the system dependencies of our case study we applied theoretical sampling in order to capture all possible variations of a dependency resolution and gain a deeper understanding. Therefore, among the selected dependencies there were dependencies resolved by control, dependencies resolved by trust, and dependencies that were not resolved and we identified and analysed trustworthiness requirements. Theoretical sampling of the system dependencies was also a way to ensure data triangulation of our case study and ensured its external validity.

Reliability focuses on whether the case study yields the same results if another developer replicates it. Triangulation, developing and maintaining a detailed case study protocol, spending sufficient time with the case ensured a certain level of similarity of results and improved the validity of the study. However, due to the fact that the trust analysis is carried out from the perspective of the developer it is very natural that the results can differ in certain aspects. In particular, when reaching the stage to decide the type of resolution of a dependency, another developer can identify the same control, normative trust, or external trust resolutions as these

resolutions are related with the system environment and they are not related to the developer. On the hand resolutions, such as experiential trust and reported trust are strongly related with the individual developer. As a result, another developer might not resolve a dependency with an experiential trust or reported trust resolution because he might not have a direct experience with the trustee or not know the reporter respectively. This can also occur vice versa, for example another developer might identify an experiential trust resolution for a dependency while we didn't because we didn't have any direct experience with the trustee.

In addition, method triangulation was used to improve the external validity. The first data collection method was observation, a direct kind of method, where there was direct contact and real time data collection. Observation was conducted in order to investigate how a certain task is conducted by a practitioner. The advantage of the observation data collection technique was that it provided a deep understanding of the phenomenon that was studied. The second data collection method was a documentation analysis of archival data. It included the analysis of work artefacts which were already available. One disadvantage of such kind of data collection technique was that the documents were created for another purpose than that of the research study, so it is not certain if the requirements on data validity and completeness were the same.

### 6.1.2 Data collection - Applying the JTrust methodology

We apply the JTrust methodology. This entails that there is already a scenario and requirements from the stakeholders. There have already been discussions with domain experts. Therefore, at this stage of the system development the JTrust methodology can be applied to model the NHS system requirements to model and reason about the trust relationships and assess the trustworthiness of the NHS system. We performed the role of the requirements analyst to carry out the following steps of the methodology. The identified trust relationships are between the requirements analyst and the various entities with which the system is interacting.

#### **ACTIVITY 1: DEPENDENCY ANALYSIS**

From the scenario, we identify the following main actors, Patient, GP, Pharmacist, NHS England, Health and Social Care Information Centre and NHS System. We carried out discussions with domain experts and studied organisation documents that were publicly available. These documents were describing the structure of the NHS in England and its policies.

- **Patient:** The patient is a person who wants to receive health care service in England (NHS, 2013b). This includes access to local services such as the GP,

hospital or clinic, or health improvement services provided by the patient's local community . In addition, the patient wants his information to be kept confidential and his privacy to be respected. This does not mean that his information will not be shared but it does mean that it will only be shared with his agreement (consent) or if there is another legal basis. Also, the patient can choose the GP practice that he would like to register. The patient can also nominate a pharmacy from where he wants to collect his medicines. The patient has an NHS number that is the only National Unique Patient Identifier, used to help healthcare staff and service providers match the patient with his health record. Everyone registered with the NHS in England and Wales has his own NHS number. The NHS number acts as the key to unlocking services such as Choose and Book and the Electronic Prescription Service. A patient should know his NHS number as this can help those treating him to find his records more quickly and share them more safely with other health care professionals.

- **NHS England:** The NHS England is an independent body, at arm's length to the UK government (*NHS England*). The main aim of the NHS England is to improve the health outcomes for people in England by commissioning primary care and specialist services and to oversee the operation of clinical commissioning groups, which include GP practices and NHS hospitals.
- **Health and Social Care Information Centre:** HSCIC is the UK national provider of information, data and IT systems for health and social care (NHS, 2013a). It supports the delivery of IT infrastructure, information systems, and standards to ensure that information flows efficiently and securely across the health and social care system, in order to improve patient outcomes. It can be directed by the secretary of state or NHS England.
- **NHS System:** It is a group of systems that offer health care services. Some of them are the following (NHS, 2013a):
  - Automatic Identification and data capture (AIDC), which is the use of machine readable codes such as barcodes and Radio Frequency ID tags. It intends to improve patient identification, medical record tracking, pharmacy services, and asset management.
  - Choose and Book, which is a national electronic referral service which gives patients a choice of place, date and time for their first outpatient appointment in a hospital or clinic.
  - Electronic Prescription Service (EPS), which enables prescribers, such as GP's and practice nurses, to send prescriptions electronically to a dis-

penser, such as a pharmacy, of the patient's choice. This makes the prescribing and dispensing process more efficient and convenient for patients and staff.

- GP2GP, which enables patient's electronic health records to be transferred directly and securely between GP practices.
  - NHS e-Referral Service, which improves the quality of the referral experience for patients and better support business processes for clinicians and administrative staff.
  - NHSmail, which is a secure email service and its purpose is for sharing patient identifiable and other sensitive information.
  - Picture Archiving and Communications System (PACS), which enables x-ray and scan images to be stored electronically and viewed on screens, helping to improve diagnosis methods.
  - Registration Authorities and Smartcards, which are used by health professionals for secure access to confidential information, as governed by registration authorities.
  - Summary Care Record (SCR) (NHS, 2011a; NHS, 2012b). A patient's SCR contains essential health information about any medicines, allergies, and adverse reactions derived from their GP record. Where a patient and their doctor wish to add additional information to the patient's SCR, this should only be added with the explicit consent of the patient. Once SCRs are created, authorised NHS healthcare staff in urgent and emergency care settings that need access to the information will view these records when delivering clinical care. Figure 6.4 shows how the SCR looks (NHS, 2012a).
  - Spine, which provides the infrastructure that enables increased patient safety, improved quality of healthcare, greater clinical effectiveness, and better administrative efficiency. Its role is to support all the previous services, by providing a central repository of SCRs that can be shared with NHS staff, a national repository of NHS organisations and all registered users, authentication of staff access, and supports a single NHS number as a unique identifier facilitating the safe, efficient, and accurate sharing of patient information across organisational and system boundaries within the NHS.
- **General Practitioner (GP):** The General Practitioner is a doctor who works in primary care (*NHS Careers*). They are the first point of contact for most

patients. The bulk of the work is carried out during consultations in the surgery and during home visits. GPs provide a complete spectrum of care within the local community, such as dealing with problems that often combine physical, psychological, and social components. GPs call on an extensive knowledge of medical conditions to be able to assess a problem and decide on the appropriate course of action. They know how and when to intervene, through treatment, prevention and education, to promote the health of their patients and families. Most GPs are independent contractors to the NHS. This independence means that in most cases, they are responsible for providing adequate premises from which to practise and for employing their own staff.

- **Pharmacist:** A pharmacist is an expert in medicines and their use (*NHS Careers*). The majority of pharmacists practice in hospital pharmacy, community pharmacy, or in primary care pharmacy, working to ensure that patients get the maximum benefit from their medicines. They advise medical and nursing staff on the selection and appropriate use of medicines. They are responsible for dispensing medicines, and clarify with GPs and other prescribers that dosages are correct. In addition, provide information to patients on how to manage their medicines to ensure optimal treatment, for example, making sure that patients are aware of potential side effects. Finally, Pharmacists can also give advice to members of the public on how to improve their health and well-being.

Once we had gathered as much information as we could from the domain, we constructed the system under development goal model. By system, we mean the information system for NHS that includes the aforementioned services that establish a paperless national health service. We do this in order to model the goals of the system under development and to identify which of them can be accomplished by the system itself and for which ones the system is depending on other entities. It was vital to examine this, as these dependencies are a potential threat to the system trustworthiness. The NHS system goal diagram was constructed using the JTrust tool and is depicted in Figure 6.5.

The high level goal of the **NHS system** is to **provide e-Health Services** to patients and health care professionals. This high level goal was decomposed into smaller and more specific subgoals, which in turn they were decomposed into smaller subgoals. Therefore, the high level goal of the system under development is to provide e-health care services. This goal has a number of subgoals as depicted in Figure 6.5. The most important goal is **Update SCR** as it is essential that the SCRs have up-to-date and accurate information about the patient. All the subgoals

The screenshot shows a web browser window displaying an NHS Summary Care Record. The patient's name is Lallie MAITLAND-EDWARDS, born 26-Aug-1944, female, with NHS number 943 002 8049. The page is titled 'General Practice Summary' and was created on 01-Dec-2008. A red box highlights the 'Time of sending' as 01/12/2008 13:55:00. A callout box points to this field with the text: 'Time and date is clearly visible indicating when the GP Practice last shared this summary'. Another callout box points to the medication tables with the text: 'Medications, allergies and adverse reactions'. The medication tables are as follows:

Repeat Medication				
Date first added	Medication Item	Dosage instructions	Quantity or duration	Reason for medication
01/12/2008	MST CONTINUS tabs 30mg	TAKE ONE EVERY 12 HRS	60 tablet(s)	
01/12/2008	DICLOFENAC SODIUM mir cap 75mg	TWICE A DAY	56 capsule(s)	
01/12/2008	PARACETAMOL caps 500mg	TAKE TWO 4 TIMES/DAY	100 capsule(s)	

Acute Medication				
Date prescribed	Medication Item	Dosage instructions	Quantity or duration	Reason for medication
01/12/2008	DIAMORPHINE HCl inj 10mg	START WITH 20MG/24 HOURS IN SYRINGE DRIVER, INCREASE AS PER PROTOCOL	10 10mg ampoule(s)	
01/12/2008	MIDAZOLAM inj 10mg/5ml	START WITH 20MG/24 HOURS IN SYRINGE DRIVER, INCREASE AS PER PROTOCOL	10 ampoule(s)	
01/12/2008	METOCLOPRAMIDE inj 10mg/2ml	START WITH 30MG/24 HOURS IN SYRINGE DRIVER AND INCREASE AS PER PROTOCOL	20 ampoule(s)	
01/12/2008	PARACETAMOL caps 500mg	TAKE TWO 4 TIMES/DAY	80 capsule(s)	

Figure 6.4: Summary care record of a patient

of the high level goal **e-Health Services** have an equal share of contribution 0.1, apart from the most important subgoal **SCR** which has 0.2. Hence, the "share" property of the decomposition links is set accordingly. The **Referral** subgoal has in turn two subgoals, the **Choose and Book** and the **e-Referral**. Both of the goals have an equal share of contribution to their parent goal, so the "share" property of their decomposition link is 0.5. Also, the **SCR** goal has subgoals as well. These are the **Create SCR**, **View SCR**, **Update SCR**, and **Additional Information**, which have an equal amount of share to their parent goal so their "share" property is 0.25. The **Additional Information** goal has two further subgoals **Get Consent** and **Add Information** with an equal amount of contribution so the "share" property of their decomposition link is 0.5. The **EPS** has **EPS Prescriber** and **EPS Dispenser** as subgoals with an equal amount of contribution 0.5. The **EPS Prescriber** has four subgoals, the **Prescription Tracker**, **Dispenser Nomination**, **Sign Prescription**, and **Cancel Prescription** with an equal amount of contribution 0.25. The **EPS Dispenser** has two subgoals, the **Download Prescription** and **Confirm Medicine** with an equal amount of contribution 0.5. In other words the system goal diagram is a hierarchy of goals of the NHS system along with their relative importance in the hierarchy. Constructing the system goal diagram is vital

in order to identify the goals that can be achieved solely by the system itself and the ones that cannot.

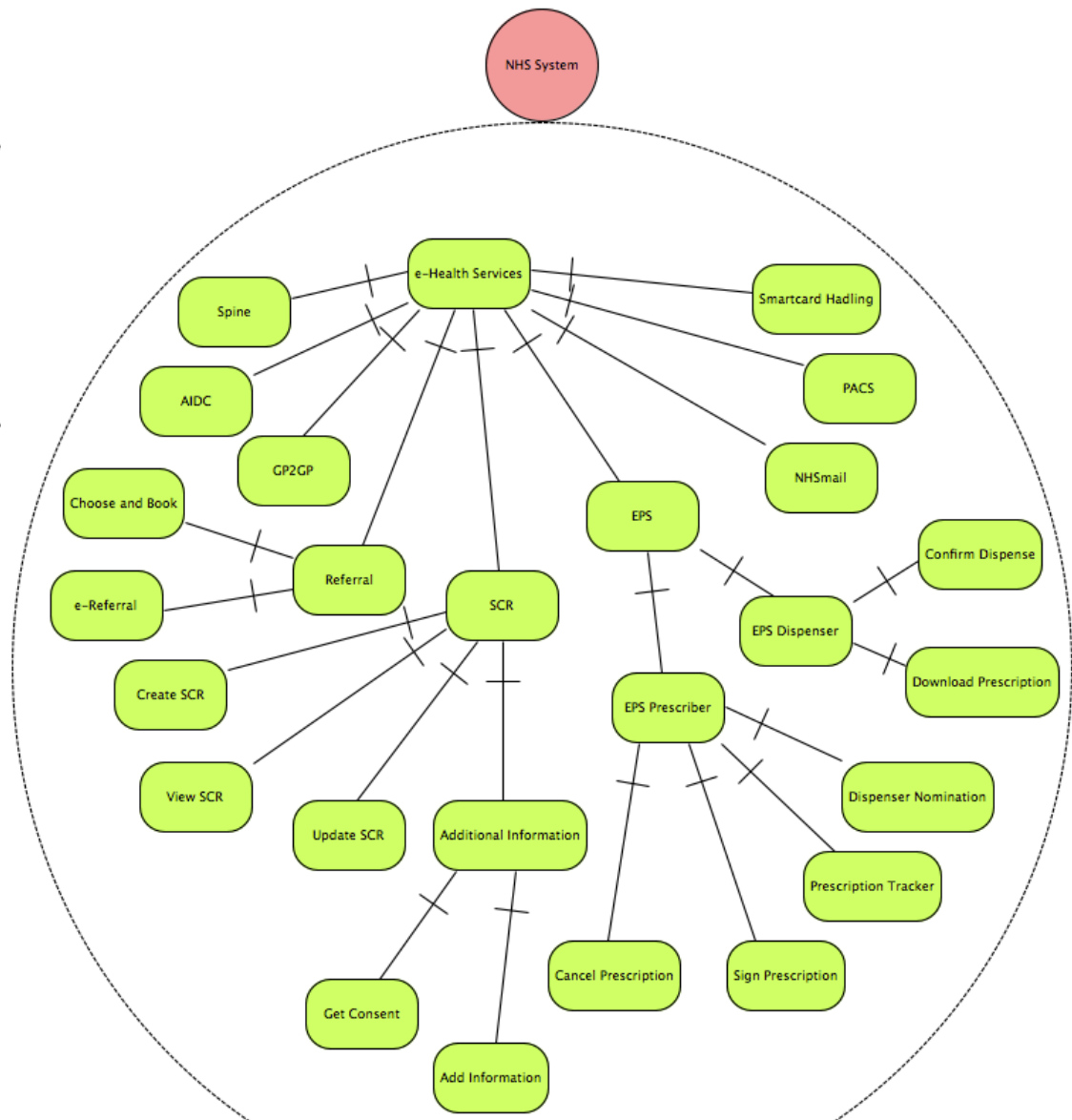


Figure 6.5: NHS System goal diagram

The next step is to examine each goal and decide whether the goals can be achieved by the system itself or it requires interacting with other actors of the system domain. Our focus is on three goals, the **Update SCR**, **Sign Prescription**, and **Confirm Dispense**, which cannot be fulfilled by the system itself unless it interacts with other entities of the system domain. The rest of the goals are considered as goals that can be fulfilled by the system. Thus, using the tool we change the capability property of the goals to false where needed (Figure 6.6). This represents

that the system cannot fulfil the aforementioned goals and the confidence level in the fulfilment of the goals is 0. For example, in order the patient's summary care record to be updated and accurate it requires the contribution from the **GP**. After a GP has examined a patient then he needs to update the patient's SCR. Only then, the NHS system can be trustworthy by maintaining updated and accurate SCRs. Up to this stage, by consulting the JTrust tool the trustworthiness of the system to be is 97.5%. Thus, for the non-achievable goals the interacting actors that contribute to the goal's fulfilment need to be identified.

Goal		
Core	Property	Value
Appearance	Capability	<input checked="" type="checkbox"/> true
	Confidence Level	1.0
	Correspondence	
	Decompositions	
	Dependency	
	Importance	1.0
	Means	
	Name	View SCR
	Super Link	+ Decomposition

Goal		
Core	Property	Value
Appearance	Capability	<input type="checkbox"/> false
	Confidence Level	0.0
	Correspondence	
	Decompositions	
	Dependency	
	Importance	1.0
	Means	
	Name	Update SCR
	Super Link	+ Decomposition

Figure 6.6: Modifying the capability property of a goal

In the system domain there are other actors that can fulfil the goals that the system is incapable of fulfilling by itself, so the appropriate correspondences need to be modelled. To this end, the goal diagrams of the candidate actors need to be constructed. For the goals **Update SCR** and **Sign Prescription** the candidate actor is the **GP**, who can update the SCR and also sign the prescription after he has examined a patient. For the goal **Confirm Dispense** the candidate actor is the **Pharmacist**. Thus, the goal diagrams for the aforementioned actors are constructed and the appropriate correspondence links are added between the corresponding goals of the **NHS System** and the **GP** and the **Pharmacist** as show in Figure 6.7. Once added, the JTrust tool automatically added the respective dependency links of the **NHS System** on the **GP** and the **Pharmacist**.

The **NHS system's** goals that cannot be accomplished by the system itself, were modelled as dependencies on other interacting actors. Figure 6.8 depicts the modelled dependences by the JTrust tool. These are the following:

- The **NHS System** depends on the **GP** to **Update SCR** after the examination of patients.
- The **NHS System** depends on the **GP** to **electronically Sign Prescription**. All prescribers need to apply advanced electronic signatures to prescriptions. These advanced electronic signatures are unique to individual users and are



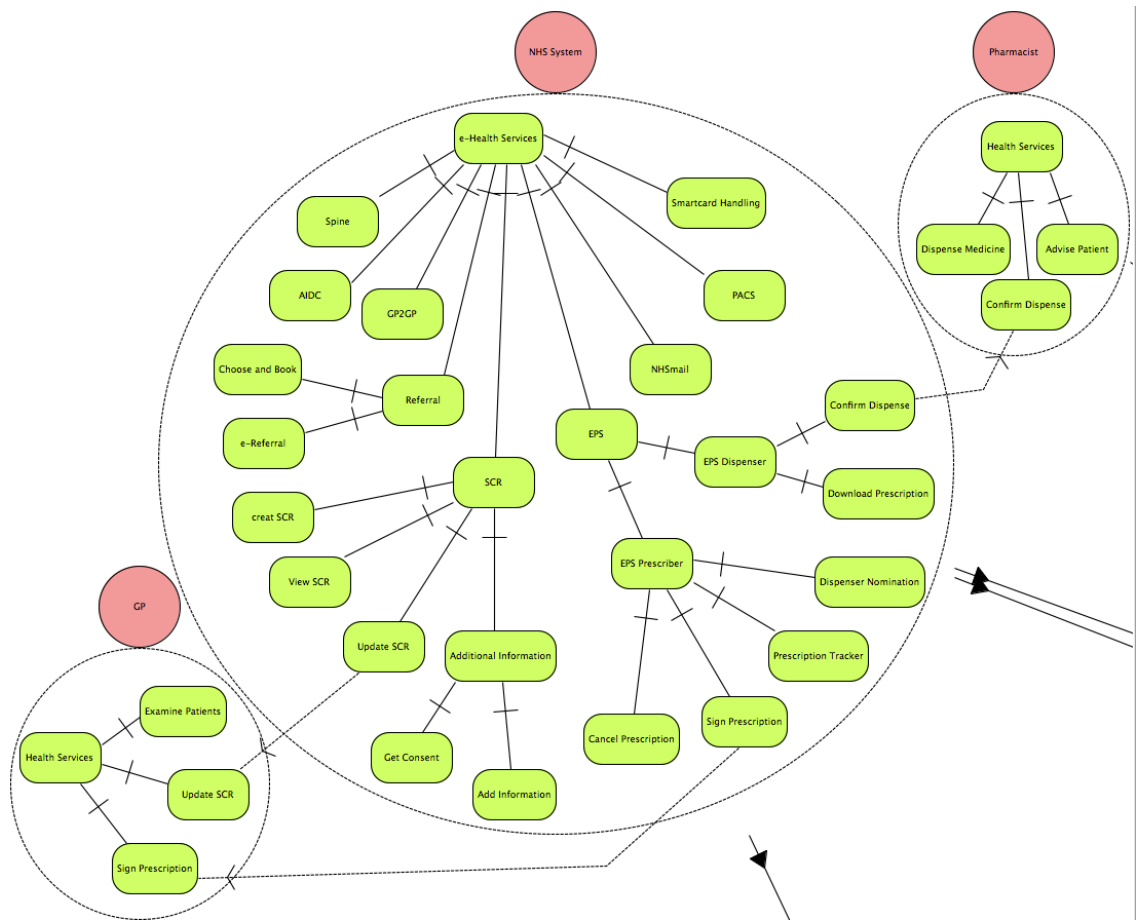


Figure 6.7: Correspondences links between NHS System and GP and Pharmacist

applied using their smartcard or passcode.

- The **NHS System** depends on the **Pharmacist** to confirm that the medicine has been dispensed to the **Patient**.

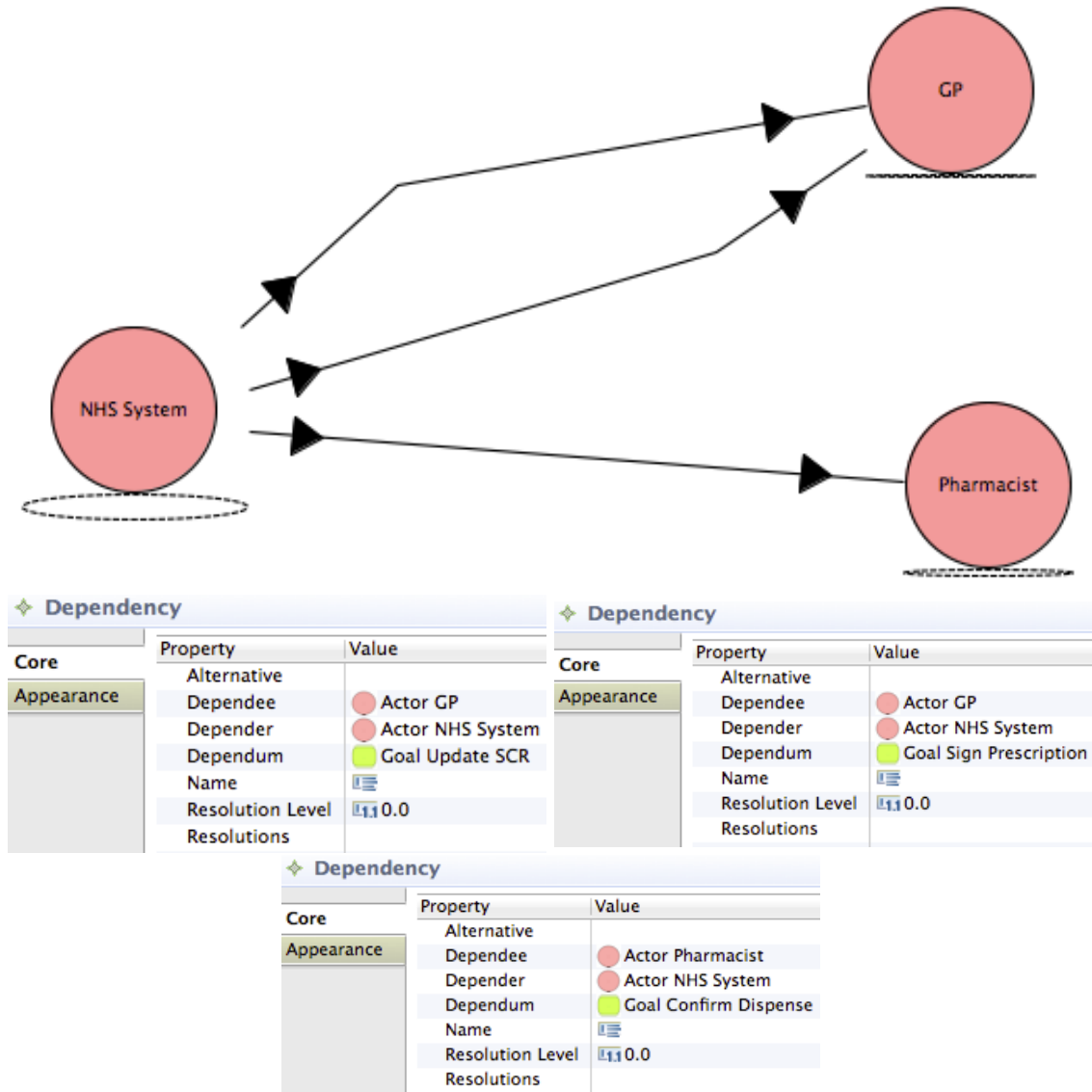


Figure 6.8: NHS System dependencies

The modelled dependencies represent the means with which the system can fulfil its goals and be trustworthy for the stakeholders. Nevertheless, they represent just assumptions, hence they require further attention.

The modelled dependencies of the system on other entities of the system domain constitute a potential threat to the NHS system trustworthiness. The system is required to be trustworthy, but if the dependencies are not fulfilled when the system

is put in operation, then it will not be trustworthy. In particular, if the GP does not constantly update the SCR then the SCR will not be updated and accurate and the system is not trustworthy in terms of holding and updated and accurate SCR. If the GP does not sign the prescription, the system will not be able to show the prescription to the Pharmacist in order to dispense the patient's medicine in time, thus the system is not trustworthy in terms of showing the prescription to the Pharmacist. If the Pharmacist does not confirm the dispensing of the medicine, the system will not have an accurate record of the medicines taken by the patient, thus the system is not trustworthy again. Moving on and implementing the system without investigating further these dependencies by identifying ways to remove the uncertainties at this stage, will result in a system that has the risk of not being trustworthy.

### ACTIVITY 2: RESOLUTION ANALYSIS

To build confidence that the NHS system dependencies on other entities will be fulfilled and the NHS system will be trustworthy, trust and control resolutions of the modelled dependencies need to be found. Each one of the identified dependencies from the previous activity constitutes a potential vulnerability to the system because there is uncertainty whether they will be fulfilled. To remove the uncertainty the dependencies must be resolved either by trust or control. The dependency on the **GP to Update SCR** is resolved by **Control**. This means that at this stage there is the assumption that the **NHS England** is controlling the **GP to update the SCR** after the examination of a patient. The dependency on the **GP to electronically Sign Prescription** is resolved by **Normative Trust**. Initial information suggests that it is a norm in the health care domain of England that the GPs sign the prescriptions in time. The dependency on the **Phamacist to Confirm Dispense** of the medicine to the **Patient** is resolved by **Normative Trust**. Similarly, it is considered a norm that the Pharmacists confirm the dispensing of medicines to the patients. For the specific dependencies the resolutions are shown in Figure 6.9. These resolutions are just assumptions at the specific stage of the software development process and they were based on initial domain information. However, the introduction of a new dependency on the NHS England raises the question of whether NHS England will actually do this once the system is implemented.

The reliance of the **NHS System** on the **NHS England** to control the **GP** introduces a new dependency on the **NHS England**. The system can be considered trustworthy as long as **NHS England** has the control means to force the **GP to Update the SCR**. As a result the System will be trustworthy because it will constantly have an updated and accurate record. Consequently, there is a new dependency, i.e., the **NHS System** depends on the **NHS England** to control the

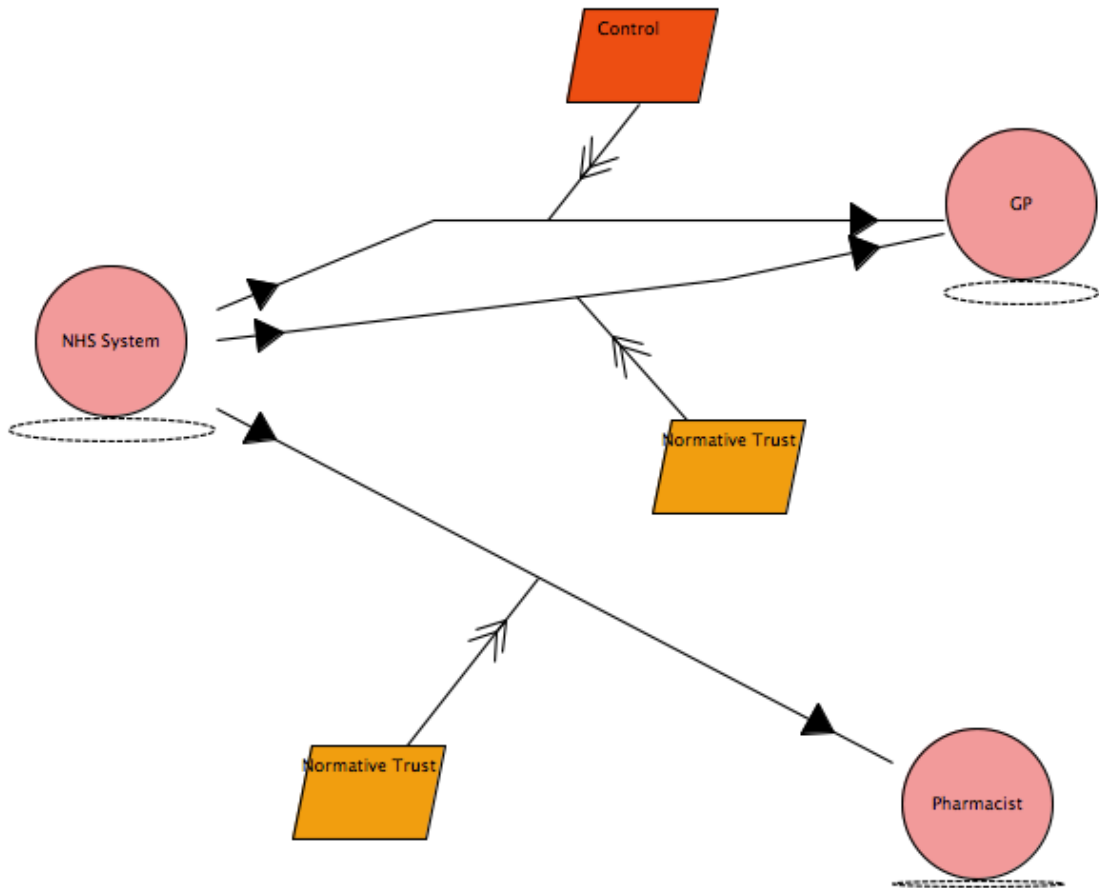


Figure 6.9: First iteration of resolution analysis

**GP**. In the developed model, the **NHS England** actor is added and its goal diagram is constructed that will contain the goal **Control GP** as shown in Figure 6.10. On the other hand normative trust type of resolutions do not introduce any new dependency.

The newly introduced dependency on the **NHS England** is a potential vulnerability for the system and needs to be resolved as well. So, the dependency is resolved by **External Trust**, i.e., the **UK Government** states that the **NHS** controls the **GP**. That is why there is no further introduction of new dependencies, but there is a need now to examine what are the entailments that derived from the identified resolutions.

### ACTIVITY 3: ENTAILMENT ANALYSIS

The identified resolutions point out the underlying trust relationships. From the resolutions we need to identify the entailments, which represent the existing trust assumptions about the trust relationships in the e-health care socio-technical system that are underlying our analysis.

The developed model for the analysis of the entailments is shown in Figure 6.11. So, the control resolution **NHS England controls GP to update SCR** required an entailment that **NHS England can was trusted to control the GP**. The external trust resolution, i.e., **UK government states that NHS England controls the GP**, required an entailment that **UK government was trusted** for what it was stating. Then the normative trust resolution that **GP signs the prescription** required an entailment that the **environment norm was trusted for GP to sign the prescription**. The normative trust resolution that **Pharmacist confirms the medicine** required an entailment that the **environment norm was trusted for the Pharmacist to do so**. These entailments would have an impact on the trustworthiness of the developed system and their validity needed justification.

The next step was to justify if the entailments were valid or not and to modify their "Valid" property accordingly as shown in Figure 6.12. The entailments by default were set as false unless we found evidence that proved the opposite. For the entailment that the **environment norm was trusted for the GP to sign prescriptions** was true as there was history of evidence that supported this assumption. However, for the entailment that **environment norm was trusted for the pharmacist to confirm the medicine given to the patient** there was evidence, i.e., history of events that showed the opposite. Therefore this is not a valid entailment and as a consequence the dependency between the system and pharmacist may not be fulfilled. In addition, for the entailment that **UK Government is trusted for stating that NHS controls the GP to update the SCR** it is false as there

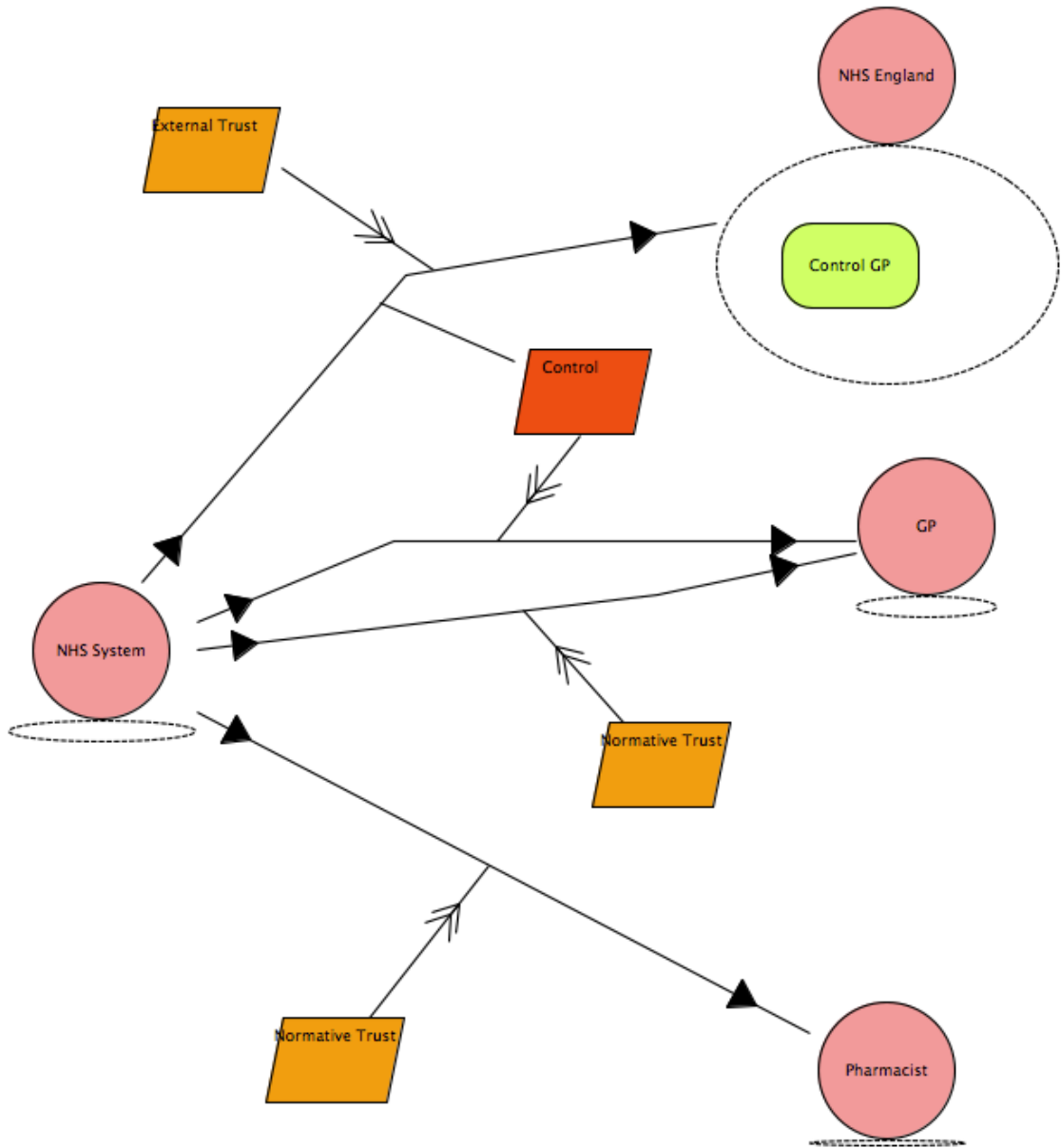


Figure 6.10: Second iteration of resolution analysis

is not sufficient evidence that UK Government is trusted in the context of health care. Therefore, the entailment that **NHS is trusted to control the GP** becomes invalid as well. The invalid entailments represented trust assumptions about trust relationships on which the further development of the NHS system would have been based. Eventually they would have become potential vulnerabilities of the system. Thus, additional requirements were added to the functionality of the system to ensure the fulfilment of dependencies and as a consequence the trustworthiness of the NHS system.

#### ACTIVITY 4: TRUSTWORTHINESS REQUIREMENT ANALYSIS

This activity considered the invalid entailments of the entailment analysis. Such entailments originated from dependencies that were not resolved and constituted a potential vulnerability to the trustworthiness of the developed NHS system, as there was no confidence that the dependencies would have been fulfilled once the system was developed and put into operation. As a result, there was the need to identify new resolutions to feel confident in the fulfilment of the dependencies. The new resolutions were control resolutions where the NHS system would act as the controller and force actors to fulfil their dependencies.

Therefore new resolutions were necessary for the dependencies between the **NHS System** and the **GP** and between the **NHS System** and the **Pharmacist** as shown in Figure 6.13. For the first case of the dependency between the System and the GP there was new system functionality added, which consisted of two trustworthiness requirements:

- i) The NHS System shall examine whether the SCR of a patient has been accessed and updated.
- ii) The NHS System shall prevent the issue of a prescription from the GP unless he has updated the patient's SCR.

Between the two requirements, the first one was observation and the second one deterrence.

For the second case, new functionality was added, which consisted of three trustworthiness requirements:

- i) A pharmacist shall input the medicine given to the patient to the NHS system.
- ii) The NHS System shall verify whether the pharmacist provides the exact medicine to the patient.
- iii) The NHS System shall prevent the pharmacist from issuing a receipt if the pharmacist does not provide accurate medicine.

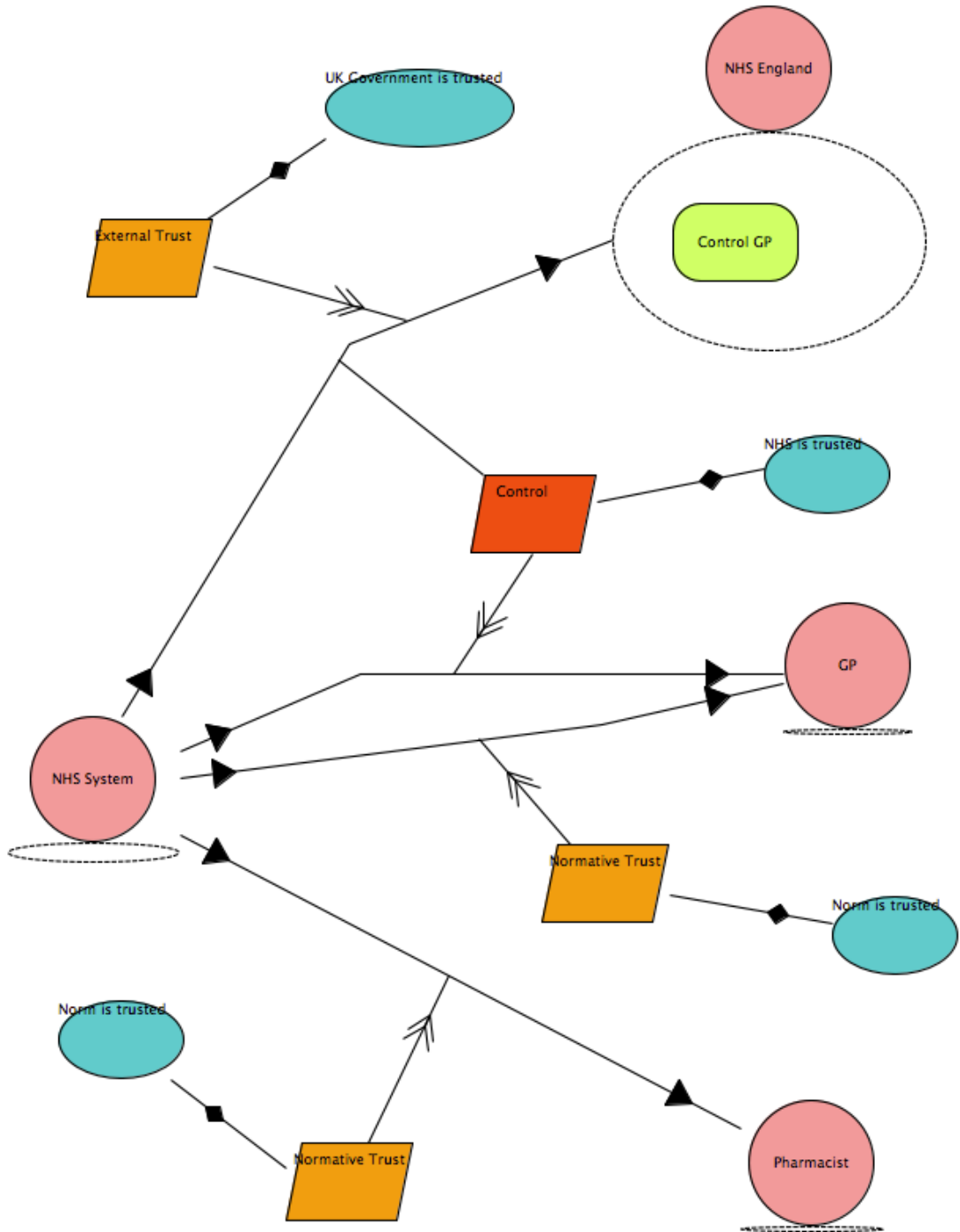


Figure 6.11: Entailments analysis



Entailment			Entailment		
Core	Property	Value	Core	Property	Value
Appearance	Name	Norm is trusted	Appearance	Name	Norm is trusted
	Valid	true		Valid	false

Entailment			Entailment		
Core	Property	Value	Core	Property	Value
Appearance	Name	UK Government is trusted	Appearance	Name	NHS is trusted
	Valid	false		Valid	false

Figure 6.12: Entailments validation

Among the three requirements, first two were observation and third one was deterrence.

Therefore, through observation and deterrence requirements the system enforces the fulfilment of the dependency on the pharmacist to confirm the medicine given to the patient and the fulfilment of the dependency on the GP to update the patient's SCR.

#### ACTIVITY 5: NHS SYSTEM TRUSTWORTHINESS ASSESSMENT

At any stage, the JTrust tool automatically assessed the NHS system trustworthiness. Before the trust analysis the trustworthiness was at 91.25% as shown in Figure 6.14. When the trustworthiness requirements were added to the system the trustworthiness became 100%.

#### 6.1.3 Data analysis - Evaluation results and discussion

Our experience in using the JTrust methodology presented in the thesis for the NHS System revealed a number of issues that are worth pointing out. The richness of the trust issues in the case study gave us confidence that we had succeeded in our aim to develop a trustworthy information systems development methodology. In general the results of the evaluation were favourable. There was a qualitative analysis of the data collected in the previous step. The objective of the analysis was to derive conclusions from the data, keeping a clear chain of evidence. The chain of evidence means that the reader should be able to follow the derivation of the results and conclusions from the collected data. To this end, sufficient information from each step of the methodology application and decision taken was presented in the previous section. During the case study we made a number of observations and these observations contributed to the evaluation of our methodology. In this regards, the main observations were the following:

- **The methodology provides a useful systematic way to identify and model direct and indirect trust relationships:** This systematic way is of

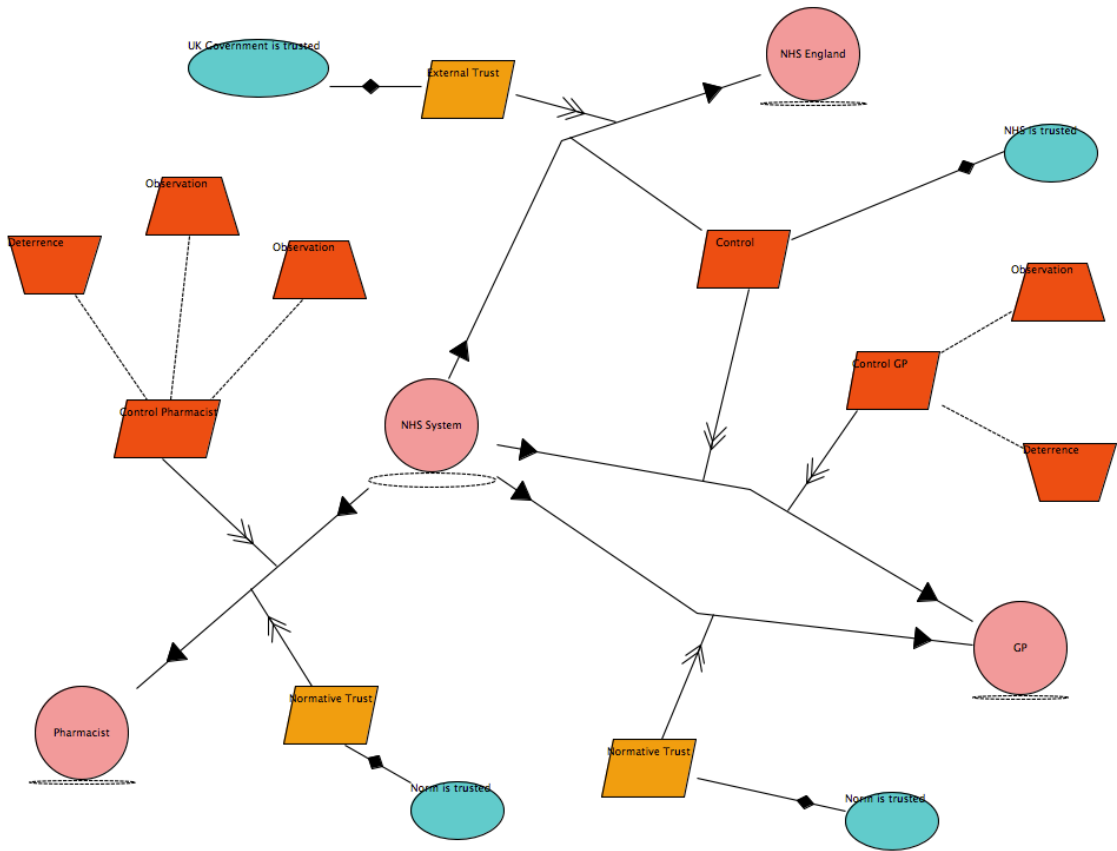


Figure 6.13: Trustworthiness requirement analysis

Actor		
Property	Value	
Core	Can	
Appearance	Name	NHS System
	Trustworthiness	91.25
	Wants	Goal e-Health Services,

Actor		
Property	Value	
Core	Can	
Appearance	Name	NHS System
	Trustworthiness	100.0
	Wants	Goal e-Health Services,

Figure 6.14: Trustworthiness level before and after the trust analysis

high importance especially when the success of the information system under development will depend on the strength of such trust relationships. Constructing a model of an information system includes modelling of entities that are free to behave however they want, such as humans and third party software. These are trust relationships that underlie the information system and the developer cannot easily identify them. The modelling of direct and indirect trust relationships makes them explicit and brings them into focus. For example, our model of the case study included a dependency on the GP to update the SCR, which was not clear to the developer that it is also a trust relationship with the GP. Moreover, the indirect trust relationship on the NHS England it would have been difficult to identify. Therefore, identifying the underlying trust relationships, both direct and indirect, is a necessary first step towards the implementation of a successful information system.

- **The methodology successfully enables the developer to reason about trust relationships:** This structured way is also important especially in situations where the development team consists of members with different cultural backgrounds and perceptions about trust. Employing JTrust's structured way of reasoning about trust relationships, i.e. experiential trust, reported trust, normative trust, external trust, it is easier for the development team to explain why they believe that a trust relationship will be realised. For example, when the model of our case study included the dependency on the GP to update the SCR and to sign the prescription, it was hard to explain why she would actually do it once the system was developed. With structure reasoning about trust relationships becomes clear where such development decisions are based.
- **The methodology successfully enables the identification, modelling, and analysis of trustworthiness requirements:** The methodology enables the developer to identify trustworthiness requirements, by recognizing functionality that is required to ensure the system trust trustworthiness and to model trustworthiness requirements by associating system functionality with specific system dependencies in which there is lack of confidence that they will be fulfilled once the system is in operation. In addition, JTrust provides a systematic way for the developer to analyse trustworthiness requirements by examining them in detail and discovering essential features, such as observation and deterrence functionalities. For example, in case of the system dependency on the GP to update the Summary Care Record there was not trust that he will do it, so a trustworthiness requirement was identified and modelled. That trustworthiness requirement was analysed and consisted of an

observation functionality to examine whether a SCR has been accessed and updated and a deterrence functionality to prevent the issue of prescription unless the SCR has been updated.

- **The methodology successfully enables the identification of situations where there is a trade-off between trust and control:** The more the developer trusts the entities that are interacting with the technical system that he is developing the less stringent the system requirements are that force the interacting entities to behave in a desired by the developer way. These situations are implicit and not clearly visible to the developer. However, employing JTrust such situations are discovered and become explicit. The developer is able to comprehend the different alternatives between trust and control and consider the benefits, risks, and implications of her decisions to the system under development. For example, in the case of system dependency on the GP to sign the prescription was resolved because there was normative trust that made us confident that the GP will actually do that once the system is put in operation. However, even if there was not enough trust the developer could have considered not to add any extra trustworthiness requirement as this would have increased the complexity of the system, decreased its usability for the GP and it would have outweighed the benefits. The introduction to the software system control measures because of lack of trust, it will have a negative effect. The control measures will result in a less usable software system and also they will decrease its performance. The negative effect is not only limited to the usability and performance of the system, but it will also affect other non-functional requirements, e.g. usability or availability.
- **The methodology successfully provides an early assessment of the trustworthiness level:** This assessment is reasonably beneficial during the early development. It provides the advantage of identifying early potential threats to system trustworthiness in order to be mitigated. Therefore, the system development is based on strong foundations that will help develop a successful system. Otherwise, the development of the system is based on the wrong assumptions and can lead to a potential failure of the developed system or cost a vast amount of resources, such as time and money, in order to fix them. For example, in the case of system dependency on the GP to update the SCR, if there was no trustworthiness requirement the GP would not have been updating the SCR, and as a result the system would not have been trustworthy because it would not possess a patient's medical record that is accurate and updated.

- **There is an overhead of modelling elements and activities:** First, when the project focuses on a large number of system dependencies, the efforts for developing and maintaining the models are considered high. In particular, JTrust needs more effort in analysing the dependencies and constructing resolutions and entailments models. The models were often becoming too big which constituted them less readable.
- **Additional guidelines are needed regarding the validation of the entailments:** The methodology identifies entailments, which are conditions of trust relationships on which the system development is based. Therefore, these entailments require validation. JTrust provides some general guidelines that evidence needs to be collected, such as historical data or testimonials from various stakeholders. However, more guidelines could have been helpful for the developer.

## 6.2 Study 2: Evaluation of JTrust methodology by survey research

An important category of empirical study is that of the survey, where expert individuals that have used specific methods or tools on projects are asked to provide information about the method or the tool (Kitchenham, Linkman, and Law, 1997). Then the information collected from the survey is analysed using standard statistical techniques. We adopted the survey method as way to further evaluate our JTrust methodology and confirm, strengthen, and further generalise our research claims about the efficacy and usefulness of our methodology.

The survey included both offline and online phases. Expert individuals, including academic and industry researchers, performed a trial of our proposed methodology using a simulated case study and then they answered the online questionnaires after an evaluation has been completed. Throughout this thesis we will use the term "user" for those expert individuals as they were the users of the methodology. The evaluation study was both a quantitative and qualitative survey. On one hand it was quantitative because it was an investigation of the number of researchers that they were agreeing with the statements presented to them. And on the other hand it was qualitative because there was a feature-based evaluation carried out by users who had studied and had experience of the methodology. Users assessed the extent to which the methodology and the tool provide the required features in as usable and effective manner based on personal opinion. The survey approach is depicted in Figure 6.15.

The goal of the survey was to build up "weight of evidence" to support our claims and confirm the propositions that were generated from the data of the previous evaluation through the case study. In particular, to collect enough data from sufficient number of subjects, i.e. the users, all adhering to the same treatment, i.e. our proposed methodology, in order to obtain a statistically significant result on the attribute of concern compared to some other treatment, i.e. the methodologies that users were using at their institution.

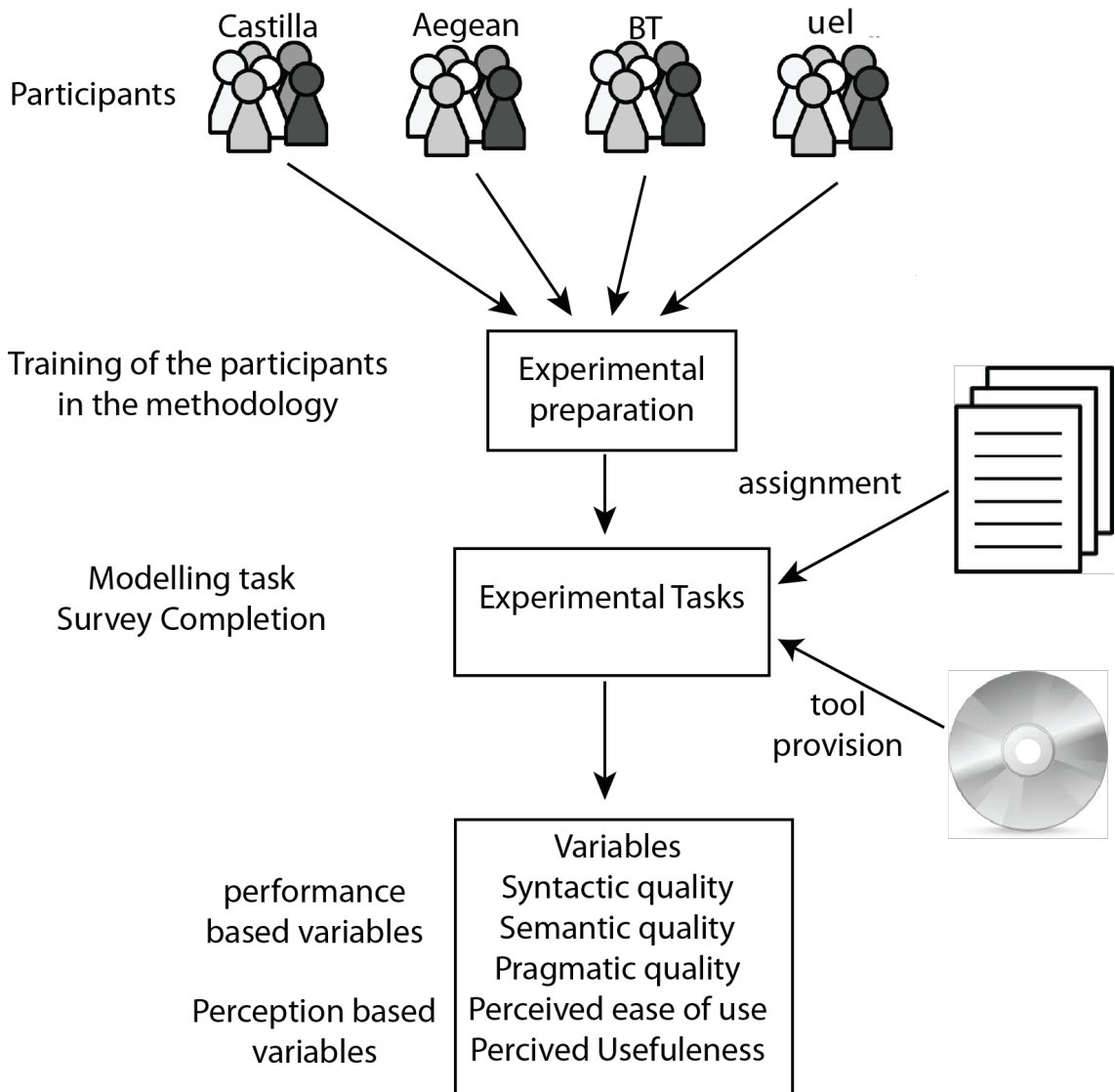


Figure 6.15: Survey approach

### 6.2.1 Survey design

#### SURVEY OBJECTIVES

The survey objectives are to answer the research questions of the evaluation study RQ4, RQ5, and RQ6. Therefore, the following research objectives were defined in order to address the research questions of the evaluation study, which used perception-based measures and performance based measures.

- O1:** Measure to what extent the methodology enables the user to model and reason about trust relationships during information systems development (perceived efficacy).
- O2:** Measure to what extent the methodology enables the user to model and analyse trustworthiness requirements during information systems development (perceived efficacy).
- O3:** Measure to what extent the methodology enables the user to assess system trustworthiness at a requirements level (perceived efficacy).
- O4:** Measure the ease of use of the methodology (perceived efficiency).
- O5:** Measure the actual effectiveness of the methodology by evaluating how well the participants perform the evaluation exercise. The evaluation will be in terms of validity and completeness of the developed models.

## **SUBJECTS**

To evaluate our methodology we organised four lab sessions, each one at a different research institution and invited groups of users to apply our methodology on a scenario. The specific institutions were selected based on judgmental and convenience sampling, because of their research speciality, which can bring more accurate results, and because of the ease of recruiting them, their accessibility to us, and the availability of limited resources. The users who participated in the sessions had already experience with software development methodologies and especially the ones from the academic institutions were familiar with virtual learning environments. In particular the lab sessions were organised at the following institutions:

- University of East London, UK. Its Software Systems Engineering Research Group is focussed on applied software engineering in an industrial context, where distributed systems engineering and secure systems engineering are essential concerns. It contributes to the fundamentals of software engineering through its research on architectures, evolution, requirements and reuse.
- British Telecom, UK. Its Security Futures Practice is conducting research in three broad areas: securing the converged network; security the virtual organisation; security management framework; In securing the virtual organisation

trust is essential in any form of communication and especially in securing relationships between enterprises. BT's researchers are exploring the development of systems that can enable the appropriate sharing of information assets while protecting them from external software attacks and physical theft.

- University of the Aegean, Greece. Its postgraduate program on cultural informatics includes a module where students learn how to analyse and design cultural systems that ensure the security and privacy of information. The students had experience with goal modelling techniques.
- University of Castile-La Mancha University, Spain. Its Institute of Informatics Technologies and Systems is to enhance research in different areas of Computer Engineering in order to develop and transfer to the organisations, information systems and technologies that contribute to the progress and well being of society. The ITSI has over 70 researchers with great experience in both basic and applied research. Researchers develop their work in eight groups addressing different research lines related to technology and information systems.

In total there were thirty-two users who voluntarily took part in the four lab sessions. More specifically, six were from the University of East London, seven from British Telecom, twelve from the University of the Aegean, and five from the University of Castile-La Mancha. Among the users were academics, industry researchers, research students and postgraduate students. Five users had more than fifteen years of experience in software engineering, seven users had experience between six to fifteen years, and the rest of the users had experience in software engineering up to five years. Also, the majority of the users were experienced in security engineering, while there were some users that had experience in trust engineering, risk analysis, software protection, privacy analysis, dependability analysis, and quality assessment. Almost, all users were familiar with UML, while there were some users that were also familiar with iStar modelling language, Secure Tropos, and PriS.

### **ETHICS PROTOCOL**

During the study the guidelines from the European Code of Conduct for Research Integrity (Drenth, 2012) were followed. More specifically, subjects were informed fully about the purpose, methods, and intended possible uses of the research, what their participation in the research entails. Also, the subjects voluntarily participated in the research study and provided their consent. In addition, the confidentiality of any sensitive information supplied by subjects and their anonymity was respected and preserved.

### **LOCATION**



The evaluation sessions were carried out at the locations of the institutions. Lab rooms were used in the case of academic institutions, while a meeting room was used in the case of BT.

### **TOOLS**

For the case of academic institutions the labs included personal computers with Microsoft Windows XP operating system, while for the case of BT the users brought their BT laptops with them. Furthermore, a web link was provided to the users in order to download the JTrust tool. The link included versions of the JTrust tool for the Microsoft Windows operating system and the MacOS operating system.

### **TRAINING**

Since JTrust is a new methodology and the users were not familiar with it, a forty-five minute presentation of the JTrust methodology was carried out in order to provide training to the users. The presentation did not include any general information regarding trust in software engineering, but it was very focused on the methodology. It explained the goal of the methodology, its concepts, the process, and the calculations regarding the resolution levels of a dependency and the system trustworthiness. At the end of the methodology presentation users' questions were answered and the scenario on which the users applied the JTrust methodology was described. The scenario was about the development of a trustworthy Virtual Learning Environment (VLE). A VLE is a system for delivering learning materials to students via the web. These systems include assessment, student tracking, collaboration, and communication tools. They can be accessed both on and off-campus, meaning that they can support students' learning outside the lecture hall 24 hours a day, seven days a week.

### **TASKS**

After the description of the scenario the users were asked to carry out a trust analysis of the system of the scenario and the following task were given to the users:

- Model trust relationships and identify the underlying trust assumptions about such those trust relationships.
- Identify and model trustworthiness requirements.
- Assess the trustworthiness of the system at a requirements level.

To this end, the users had to construct the respective JTrust models and the duration for the completion of the task was one hour. Moreover, specific functionality regarding the VLE was given to the users in order not to spend time on the system requirements but focus the trustworthiness aspect of VLE analysis. In particular, students, as users of the VLE, should be able to carry out the following actions:

- Access any academic material related to their course.
- View their coursework group members and be able to collaborate.
- Access any administration forms or informational documents related to their studies.
- Access and modify the personal information of their student record.
- Access their academic student record (grades, attendance, registered courses).

### **DATA COLLECTION METHODS**

In this study the unit of analysis was the individual user, and based on this the questionnaire was selected as one method for collecting data as shown in Figure 6.16. After completing the tasks the users were asked to complete a questionnaire. The questionnaire collected both quantitative (close questions) and qualitative data (open-ended questions). More particularly, the questionnaire contained twenty-three questions, which can be categorised as follows:

- User's profile related.
- Modelling language related.
- Tool related.
- Methodology activities related.
- Methodology usability related.
- Open recommendations.

The survey was carried out with an online questionnaire by employing the Google Forms service. Therefore the data was collected and stored online automatically.

In addition, a second way to collect data was observation of the subjects while they were carrying out the requested tasks. The observation was in terms of how much time the users were spending on each task, the questions they asked, and the comments that they made verbally. Also, the users were told that we were there for answering their questions and not to observe them in order to ensure that those being observed are not constantly thinking about being observed and to ensure that the observed behaviour was normal, i.e. what usually happens in the environment being observed and is not affected by the observer. The gathered data from the observations were recorded in the form of field notes, which begun during the actual observation, where we wrote what was necessary and we filled in the details later, but as soon as possible.

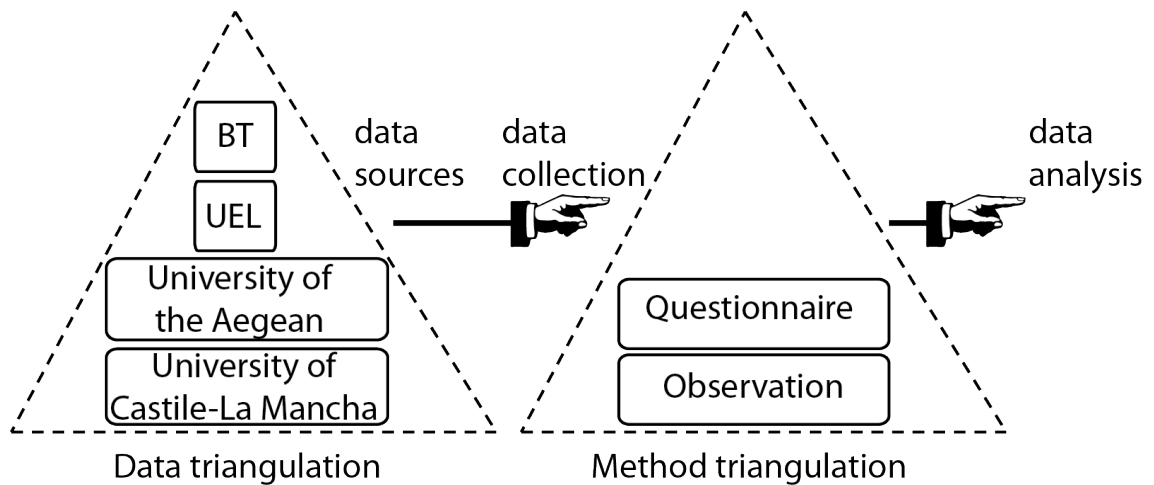


Figure 6.16: Survey data collection methods

### DATA ANALYSIS METHODS

The first step in the analysis of the survey data was to tabulate them and then to identify patterns in the tables. The second step was to evaluate the different aspects of the methodology based on the survey data. In particular, there was a distinction of two types of measures:

- Performance based measures. How well did the users perform the task? There are three measures used to evaluate the performance on the task, i.e. syntactic quality, semantic quality and pragmatic quality.
- Perception based measures. How useful and how easy to use did the users perceive the methodology? Perceived usefulness is defined as the degree to which the participant believes that the methodology is effective in achieving its objectives. Perceived ease of use is defined as the degree to which the participant believes that the methodology is free of effort.

### STUDY VALIDITY

In terms of construct validity the quality of the questionnaires was improved after several iterations of improvement by following the feedback from questionnaire experts from BT in order to make sure that there were no major discrepancies of understanding by the users. In addition, we also carried out a pilot test at the university of East London and there was a process of reflection and redevelopment of definition, in order to make sure that the users understood the terms being used. Moreover, there was an investigation in the literature and comparison with other metrics used in similar surveys. We evaluated the JTrust methodology and correctly measured quantitative metrics, such as the efficacy and the effort required to

undertake trust analysis activities in a software project. The participants answered most of the questions apart from those that were perceived as difficult at an early stage.

In order to increase the internal validity of the study we were not directly involved in the application of the methodology on the scenario given apart from answering any questions of the users regarding the methodology and observing their behaviour. Also, there was method triangulation as we collected data with two different ways, through questionnaires and observation of the users while applying the methodology.

The workshops were held at three different countries, UK, Spain and Greece in order to increase the external validity of the study. Also, participants were from various countries around the world and we ensured data triangulation. Moreover, among the users were academics, industry researchers, research students, and post-graduate students in order to ensure the representativeness of the user population.

### 6.2.2 Data collection

On the days in which the lab sessions were held, the presentation slides, the JTrust tool were made available to the users. The tool contained an already started project for the VLE scenario with an instance of a dependency resolution in order to make the users get started.

The users once they attended the JTrust presentation, started performing the evaluation tasks. We were present during this time in order to observe and capture first-hand behaviour and interactions for data collection, and to answer any possible additional questions that the users might have. The goal was to capture if possible the users' thought process and what work takes place inside the users' head. To this end we wanted to establish a communication with the users, since users reveal their thought process most naturally when communicating with other users (Seaman99), so this communication offers the best opportunity for us to observe the application of JTrust. All users constructed the JTrust models with at least a part of the system functionalities proposed.

At the of the trust analysis of the scenario, the users were given web links to the online questionnaire in order to fill in and complete the evaluation lab session. With regards to data validation, in general we observed that the users developed the proposed JTrust models satisfactorily. This means that the JTrust methodology was applied correctly and in accordance with the planning. Therefore, we can claim that the obtained data was valid to conduct the proposed evaluation.

### 6.2.3 Data analysis - Evaluation results and discussion

The results were evaluated by a combination of quantitative and qualitative analysis techniques. For the quantitative data descriptive statistics techniques were used, while for the case of qualitative data there were first coded, then patterns were identified, and generalisations could be formulated. These analyses achieved the objectives that were set at the start of the survey evaluation.

Among the users, 92% that said that the modelling language is powerful enough in order to support trust analysis and modelling. 54% of the users said that the modelling language did not included redundant concepts. Nevertheless, there was a 35% that agreed that the modelling language had redundant concepts. In addition, 65% agreed and 31% strongly agreed that the modelling language concepts were well defined respectively. Moreover, 58% of the users agreed and 27% strongly agreed that the graphical notation employed by the modelling language were intuitive (Figure 6.17).

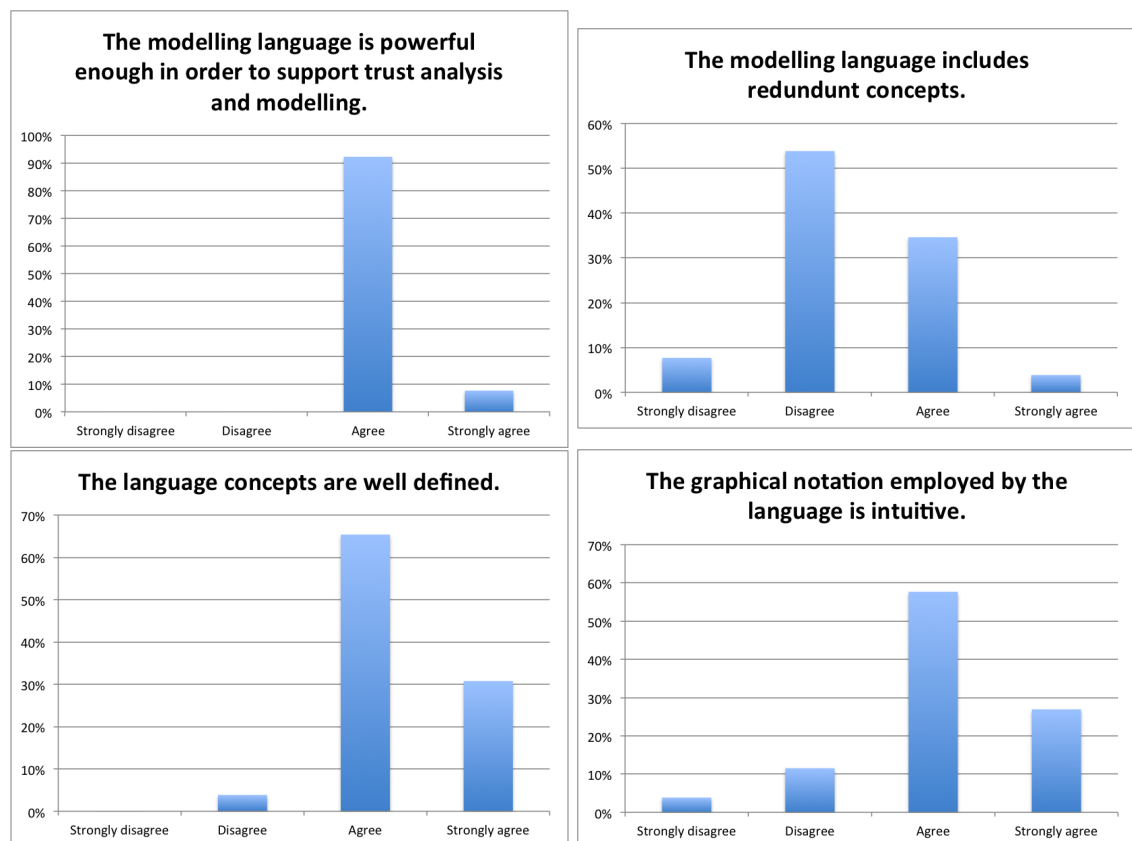


Figure 6.17: Results related to JTrust modelling language

With regards to the JTrust modelling tool, the majority of the users agreed that the tool required further improvement. On the other hand, 81% and 15% of the

users agreed and strongly agreed that the trust related functions of the tool are satisfying respectively. Moreover, 62% and 31% of the users agreed and strongly agreed respectively that the JTrust tool was easy to use (Figure 6.18).

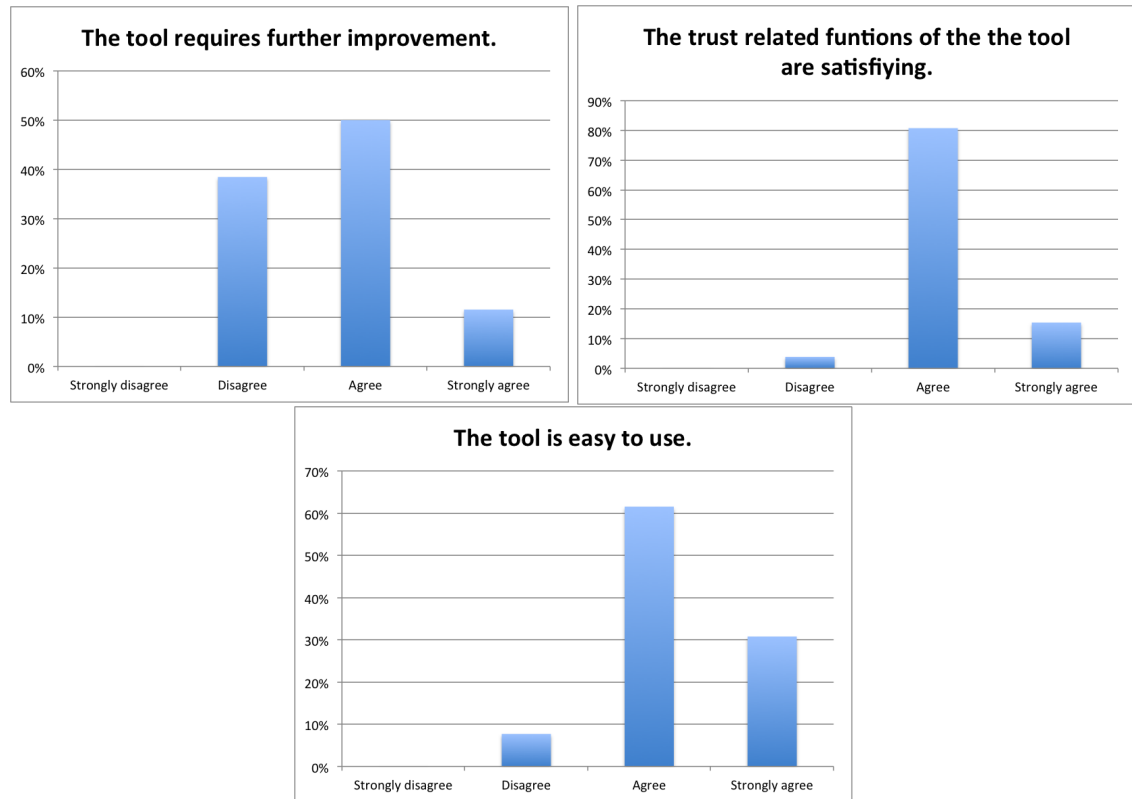


Figure 6.18: Results related to JTrust tool

With regards to the methodology as a whole, 69% and 23% of the users agreed and strongly agreed respectively that with the use of JTrust methodology the trust assumptions are explicitly captured. In addition, 50% and 46% of the users agreed and strongly agreed respectively that the methodology successfully captures trustworthiness requirements. This is a very significant result as this is one of the most important aspects of the methodology. Furthermore, 62% and 31% agreed and strongly agreed respectively that the methodology successfully assesses the system trustworthiness at a requirements level. In terms of the usability of the methodology, 58% and 35% of the users responded that the activities of the methodology were easy to follow (Figure 6.19).

The qualitative data from the users' responses were coded and patterns were identified. Based on these patterns we can report the following:

- **The trust and control techniques are useful.** In particular the techniques are useful in order to identify the nature of trust and level of control that is

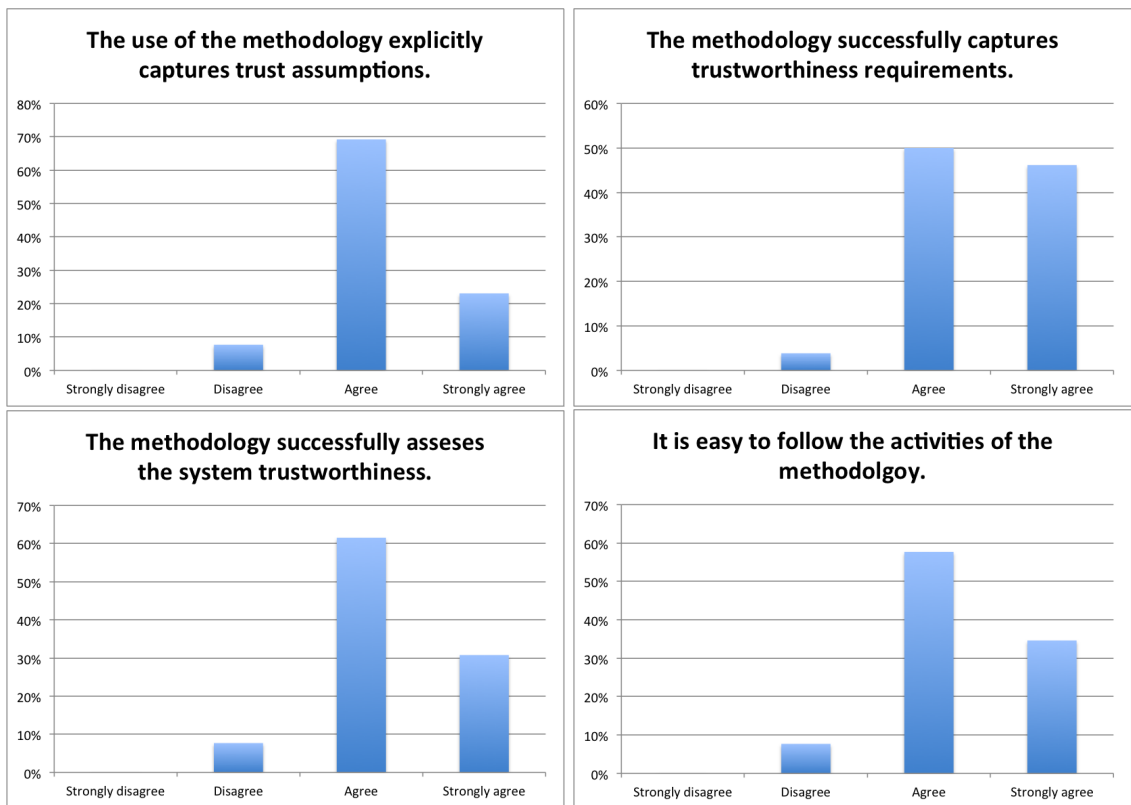


Figure 6.19: Results related to JTrust methodology

required. Moreover, these techniques enable the calculation of the system trustworthiness at a requirements level.

- **UML style of notation would be more preferable.** The goal modelling language is not very popular among developers. Thus, a JTrust modelling language with a UML-like graphical notation could have been easier to learn and to remember.
- **Developers with no prior experience of goal modelling might find challenging to use the methodology.** As the methodology is greatly based on goal modelling, it requires a decent level of knowledge of goal modelling from the developers. As a consequence, developers not familiar with goal modelling will require detailed training to learn to use the methodology.
- **The JTrust tool could be improved.** The improvement could be in the form of integration of the tool with other tools or the integration of other relevant concepts, such as associative risk, into the JTrust tool. Moreover, JTrust models tend to become large in size and there is lack of modelling space for the developer.

During the lab session we observed the execution of the survey tasks by the users and we collected data in the form of notes. This data was coded and we made a number of observations, which contributed to the evaluation of our methodology. In this context, the main observations were the following:

- **JTrust process:** The activities of the JTrust methodology were identified as fully operational and adequate, as the approach provided adequate techniques to model and analyse trustworthiness requirements. The combination of these techniques was treated as being systematic, reasonably applicable and in particular, reducing the bias for the trust relationships reasoning.
- **Artefacts:** JTrust provides graphical representation of artefacts produced by its activities. In particular, visual presentation is provided through trust models, which made it easy, to communicate the trust analysis information with other developers, as observed during the lab sessions.
- **The size of JTrust models may become big:** This is a problem the Jtrust models have inherited from goal modelling. A system under development may have a great number of interactions with other entities of the system environment. By representing all the system dependencies in a model the model becomes large in size. JTrust adds an overhead to that, since it requires the



modelling of resolutions and entailments of dependencies. Also, it requires the modelling of trustworthiness requirements. As a result, the models tend to become big and complex to read.

- **Entailment validation techniques were limited:** We have proposed that once the entailments have been identified and model, the developer needs to seek evidence, such as historical records and past experience, in order to examine their validity. However, this may not be adequate for the developer in order to proceed to such examination of validity. To this end, new techniques could be needed. For instance, for each entailment there could be techniques that guide the developer on where to seek for evidence and what kind of evidence to seek. Such techniques will allow a more system validation of the entailment and will complement the methodology.

### 6.3 Chapter summary

In this chapter we have conducted an empirical evaluation of our proposed methodology. It included two evaluation methods, a feature based case study, which concerned the e-health care system in England, and a qualitative and quantitative survey. For the survey we organised four lab sessions inviting software developers and researchers to apply our methodology. The case study and the survey enable us to evaluate our methodology. We have reported the feedback we got from the software developers and the researchers and the observations we made during the case study and the lab sessions. This included the limitations of our methodology and the areas that need further research. Overall, the JTrust methodology was found to be valid, and the observations during the case study were consistent with each other and with the feedback that we got from the users. Our findings were that our proposed methodology was both easy to use and useful in modelling and reasoning about trust relationships, modelling and analysing trustworthiness requirements and assessing the system trustworthiness at a requirements level.

## Chapter 7

# Conclusions and future work

Trust is so all pervasive in all of our lives, online or not, that it sometimes seems strange to either have to ask or answer the question of why trust is important. Trust is a part of the decision making process and as such it remains a paramount part of our daily lives, which is especially true when other humans are involved when making decisions. Such is the case of the development of an information system where decision-making is at the core of the process and where trust is an element in the decisions. Real world systems involve large populations of humans who use, configure, and maintain them. The trustworthiness of the information system depends on the trust relationships between the developer and the components of the information system. The components are expected to behave in a desired way in order the system to meet all its goals. This behaviour is reflected in the model the developer is constructing during the requirements stage.

However, components do not always behave as expected. For example, they tend to circumvent, misuse, and abuse security controls. They do not always do it as attacks, but as a way to achieve their job activities and organisational goals. Therefore, appropriate constructive techniques are required to enable the developer to capture trust relationships and to reason about them. Also, constructive techniques are required to enable the developer to identify trustworthiness requirements. To this end, this thesis aimed to develop appropriate trust and control abstractions, along with a methodological process that provides such ability to the developer. In addition, it enables her to assess the trustworthiness of the system to be a requirements stage.

## **7.1 Thesis summary**

In this thesis, we have proposed a requirements engineering methodology for modelling and reasoning about trust relationships and modelling and analysing trustworthiness requirements. We adopted Goal Oriented Requirements Engineering concepts, such as goal, actor, and dependency, in order to capture the trust relationships between the developer and the components of the information system under development. Such trust relationships concern the behaviour of the system components and they affect its ability to meet its requirements and be trustworthy. So capturing those trust relationships with the system components and their expected behaviour is essential for achieving system trustworthiness.

We suggested to resolve the uncertainty of the behaviour of the system component by utilising means of building confidence in their expected behaviour. To this end we defined trust and control abstractions in order to enable the developer to reason about her trust relationships and build confidence. Sources of trust were categorised in four categories, Experiential Trust, Reported Trust, Normative Trust, and External Trust. On the other hand, control components that make control effective were defined such as Observation and Deterrence. Reasoning about trust relationships allows the assumptions that underlie the system development and can potentially harm its trustworthiness if they are not valid to surface naturally. All the end assumptions are assumptions of trust. Once identified, the developer can examine their validity and proceed to a justified decision about the design of the system.

Furthermore, constructive techniques were developed for the modelling and analysis of trustworthiness requirements. In situations where trust assumptions are not valid, the software system-to-be acts as the controller who enforces system components to comply with the expected behaviour. To this end, the control abstractions are used in order to identify the capabilities of the system to observe certain resources for monitoring purposes, and the deterrence capabilities, for preventing system components from achieving some goals of their own, as a leverage for enforcing them to behave in an expected way.

Moreover, algorithms were developed for evaluating the trustworthiness of the system at a requirements stage. The first algorithm calculates the resolution level of a dependency, while the second algorithm calculates the confidence level of a goal. The third algorithm calculates the trustworthiness of the system under development by considering the confidence level of the high level goals of the system along with their importance to the overall system trustworthiness.

We have developed a CASE tool, named JTrust Tool, which supports the devel-

oper in using the methodology. Trust models can be constructed using the modelling capabilities of the tool and also the syntactical correctness of the models is automatically checked by utilising the meta-model. Moreover, the developed algorithms are run by the tool, so the resolution levels of a dependency, the confidence level of goals, and the system trustworthiness are automatically calculated by the tool.

We have applied our methodology using a case study from the e-health care sector in England. The case study was an e-health system for storing medical records of patients, which allows the access of basic patient information from any hospital or medical care record in England. It enables also a more efficient dispense of medicines to patients. Furthermore, a survey was conducted in four different institutions across Europe. Three of these were universities, University of East London, University of the Aegean, and University of La Mancha, while one was a company from industry, the British Telecom. The methodology was evaluated based on the results of the case study and the survey, and the evaluation results were discussed.

## 7.2 Research questions

A set of research questions was formulated according to two phases, the development of the methodology and the evaluation of the methodology. In this section we review the research questions and the findings, which have been presented as initial answers to them. In Table 7.1 we present a summary of them and then we explain each row.

- RQ1: What are the required trust abstractions and their relationships in order to reason about trust relationships at a requirements stage? We defined four trust abstractions, namely experiential trust, reported trust, normative trust, and external trust. We defined their relationships in a semi-formal way by defining a meta-model that describes the relationships with each other and with concepts adopted from goal oriented requirements engineering.
- RQ2: What are the required abstractions and their relationships that can ensure the development of trustworthy information systems at a requirements stage? We defined two control abstractions, namely observation and deterrence. Similarly, we defined their relationships in a semi-formal way by defining a meta-model that describes the relationships with each other and with concepts adopted from goal oriented requirements engineering.
- RQ3: How can we assess trustworthiness of the system under development at a requirements stage? The ability of the system and the resolution levels of the dependencies inside the information system are required to be considered. We

Table 7.1: Summary of research questions and answers

<b>Research Phase</b>	<b>Research Question</b>	<b>Research Answer</b>
Methodology Development	RQ1: What are the required trust abstractions and their relationships in order to reason about trust relationships at a requirements stage?	Trust abstractions, JTrust meta-model, and JTrust Process (Sections 3.5, 3.7, Chapter 4)
	RQ2: What are the required abstractions and their relationships that can ensure the development of trustworthy information systems at a requirements stage?	Control abstractions, JTrust meta-model, and JTrust Process (Sections 3.5, 3.6, 3.7, Chapter 4)
	RQ3: How can we assess trustworthiness of the system under development at a requirements stage?	Trust and Control abstractions, JTrust meta-model, and JTrust process (Sections 3.5, 3.6, 3.7, 3.8, Chapter 4)
Methodology Evaluation	RQ4: How well does the methodology support modelling and reasoning about trust relationships?	Case study and survey evaluation (Chapter 6)
	RQ5: How well does the methodology support trustworthiness requirement modelling and analysis?	Case study and survey evaluation (Chapter 6)
	RQ6: How well does the methodology assess the system trustworthiness at a requirements level?	Case study and survey evaluation (Chapter 6)

developed two algorithms that calculate the confidence and resolution level of goals and dependencies and a third algorithm that calculates the overall system trustworthiness by considering the confidence and resolution levels.

RQ4: How well does the methodology support modelling and reasoning about trust relationships? During the case study we observed that the methodology fully supported the modelling and reasoning of trust relationships. Furthermore, 92% of the participants responded the same thing in our survey.

RQ5: How well does the methodology support trustworthiness requirement modelling and analysis? During the case study we observed that the methodology fully supported the modelling and analysis of trustworthiness requirements. Furthermore, 46% and 50% of the survey participants strongly agreed and agreed respectively with our observation.

RQ6: How well does the methodology assess the system trustworthiness at a requirements level? During the case study we observed that the methodology successfully assesses the system trustworthiness at a requirements level. Furthermore, 31% and 62% of the survey participants strongly agreed and agreed respectively with our observation.

### 7.3 Contributions to the state of the art and impact

In the literature, there were gaps in the reasoning of developers' trust relationships with components of an information system and in the identification and analysis of trustworthiness requirements. The research that was carried out in this thesis contributed immensely towards filling this gap and allow for reasoning of trust relationships and modelling of trustworthiness requirements. In section 1.4, the four novel contributions of the thesis were briefly introduced, along with the development of a CASE tool. However they can be grasped in more detail as to how fruitfully they have contributed to the state of the art from the answers to the research questions and more precisely they were the following:

**Contribution 1: The identification of limitations, problems, and challenges of the current state of the art with respect to trustworthy information system development.** This contribution provided the foundation for the research carried out in this thesis. This contribution substantiates that there are limitations in the current state of the art.

**Contribution 2: The development of trust abstractions as part of a modelling language and their associated constructive methods to allow**

**modelling and reasoning of trust relationships.** This paper introduces a trust based process that enables the developer to identify direct and indirect trust relationships and to analyse the respective trust assumptions. The process makes use GORE, trust, and control related concepts. Using concepts like resolution and entailment allows to model and reason about the trust relationships. Therefore, trust assumptions within a socio-technical system are not left unexamined by using our approach. The proposed process leads to design trustworthy system. Nevertheless, a trustworthy system does not ensure that users will fully trust the system. User's trust is subjective and depends on factors like marketing of the final product and reputation. However, the trust abstractions establish a common understanding of trust among the developers. There is a clear definition about trust and trustworthiness, so the developers are not confused from the subjectivity of the concept.

**Contribution 3: The development of control abstractions as part of a modelling language and their associated constructive methods to allow modelling and analysis of trustworthiness requirements.** Trustworthiness requirement is the functionality added to the system in order to replace gaps of trust in the chain of trust relationships between the system and its environment. A modelling language for trust and control in order to represent how confidence in the dependencies between the system under development and its environment is achieved. A process, which describes the methods in a structured way that constitutes the development tasks clear and visible to the developers.

**Contribution 4: The development of a methodological process that employs the modelling language in order to reason about trust relationships and identify trustworthiness requirements in a systematic and structured way.** Moreover, the development of algorithms that allow the early assessment of the trustworthiness of the system under development at a requirements level.

**Contribution 5: The development of a CASE tool to support the proposed methodology. The development of a supporting tool for the methodology was deemed of high importance.** Therefore, a CASE tool was developed to support the developers in constructing the required models to analyse the trust relationships. It enables the developers to model dependencies among actors, their kind of resolutions, i.e., experiential, reported, normative, and external trust, and control, along with their respective entailments. The tool is used throughout the case study. The case study was from the health care sector where the applicability and benefits of the proposed process were demonstrated. In particular, the case study revealed that the initial design is based on trust assumptions that were not valid. As a consequence it could lead to an untrustworthy system. Using

the proposed process and concepts, those invalid trust assumptions were identified and extra functionality was added to the system design to address the identified vulnerabilities.

## 7.4 Limitations of the approach

While our proposed methodology enables the developer to model and reason about trust relationships, model and analyse trustworthiness requirements, and assess the system trustworthiness at a requirements stage, this ability is also limited in some aspects. The following list provides some points of criticism that we have identified for the approach:

- Castelfranchi and Falcone (2000) argues that the measurement of trust should not be attempted because of the complexity of the process of trust and they multiple aspects that it has. Nevertheless, still when a phenomenon is simplified it provides great insights to an issue that in other circumstances it will not be investigated (Cofta, 2007). We believe that the proposed trust and trustworthiness metrics, no matter how simplified they are and they may not exactly measure trust and trustworthiness, are still useful. They force the developer to look into issues that otherwise would not have considered. The investigation of whether the dependencies will be fulfilled once the system is put in operation or what is required in order to ensure their fulfilment is necessary in order to build strong foundations from the requirements stage and proceed to further system development. Trust levels are ambiguous at best, in terms of semantics and subjectivity, but we will use them anyway. We believe benefits far outweigh their disadvantages, and include the ability to narrow down and discuss important aspects.
- Trust cannot give us certainty, it is a judgement based on evidence, (Marsh and Briggs, 2009). Active components can still go against a control and they can still go against the expected behaviour (Möllering, 2005). In other words this means that even if there are control measures in place for an individual to behave as expected, she can still behave in a different way. This is apparent in everyday life where even though there are prisons as a control measures for individuals that commit illegal acts, yet they still commit illegal acts. Going back to the running example, even if there is system functionality that controls the lecturer to upload the lecture slides in advance of the lecture, she may still decide not to do so no matter how strong is the control measure.



- The trust analysis is occurring during the development stage. Over time things can change, such as norms and entities may not be trusted anymore. Trust relationships change over time and therefore the trust analysis made during the development time may not be valid anymore and constitute the system not trustworthy to meet all its requirements. For example, if during the development of the system it was decided that the lecturer can be trusted to upload the lecture slides in advance of the lecture this trust relationship may change after some time and the lecturer may not be trusted anymore. By that time though the system could be already developed and put in operation. Thus, it may not be trustworthy in terms of providing the lecture slides to the students because the lecturer is not uploading the slides and there is no control measure to force him to do so.

## 7.5 Future work

For the future work, we are interested in addressing some limitations of our proposed methodology and some further challenges that have appeared in developing trustworthy information systems, and propose possible areas for future research that build upon the contributions arising from this thesis. The research questions are the following:

- What are the requirements that lead to adaptation capabilities of a system because of changes in the trust relationships? A more dynamic assessment of trust assumptions is required in order to adapt to possible changes. Moreover, modern software systems become more complex and the environments these systems operate in, become more and more dynamic. So, users' expectations change and this has an immediate effect on users' trust level (Hoffman, Lawson-Jenkins, and Blum, 2006). Also, the number of stakeholders increase and the stakeholders' needs change constantly as they need to adjust to the constantly changing environment. A consequence of this trend is that the number of requirements for a software system increases and changes continually. As a result, in order for software systems to maintain their ability or competence as trustees to satisfy their trustors, which are the stakeholders, and remain trustworthy need to adapt to the changes.
- What other concepts can be used that motivate entities to behave as expected? Apart from control there are also other reasons that someone can show trustworthy behaviour. Such motives can be ethical obligation or driven by greed (Cofta, 2007). Instead of deterring a system component from achieving one of

her own goals an alternative way to motivate her to behave in an expected way is to provide a reward as long as she behaves in an expected way.

- How to engineer systems that facilitate the assessment of trust among parties? This research questions belongs in the area of trust management. Transacting and interacting through information systems and computer networks makes it difficult to use traditional methods for establishing trust between parties, such as the ones used in face to face communication between humans. Modern information systems are increasingly removing us from familiar styles of interacting and doing business, which both rely on some degree of trust between the interaction or business partners. Most traditional cues for assessing trust in the physical world are not available through information systems. So, there is the need for systems that enable the assessment of trustworthiness of another party that someone is collaborating with. Systems that enable trustworthy parties to demonstrate their trustworthiness, while they prevent parties that are not trustworthy to show themselves as trustworthy.

While the tasks for future work are the following:

- Develop validation techniques for entailments. This requires more research in human psychology and will still be far from perfect. As future work, we intend to propose methods that will further support developers for validating the entailments, for instance what kind and how much evidence is required for the entailments to be validated.
- Formalise the proposed methodology. The formalisation will complement the methodology by extending the JTrust modelling language into a formal specification language. It can also be employed to perform formal analysis of the system and verify the trust model by employing formal verification techniques.
- Improve the CASE tool. Trustworthiness requirement concepts should go inside the system goal diagram, because they are system functionality. Also improvements in terms of usability, such as to not allow the user to put whatever values in the share property of the decomposition link and add colours to the graphical notations to represent different value properties.

Apart from the specific research questions and tasks, another interesting area for future research is computational models of trust that can be used to create substitutes by which software agents can derive trust in others through information systems. The developed trust abstractions could be incorporated into such computational models of trust.



# Bibliography

- Antón, A. I. and C. Potts (1998). “The use of goals to surface requirements for evolving systems”. In: *Proceedings of the 20th international conference on Software engineering. ICSE '98*. Kyoto, Japan: IEEE Computer Society, pp. 157–166.
- Avizienis, A., J.-C. Laprie, and B. Randell (2004). “Dependability and Its Threats: A Taxonomy”. In: *Building the Information Society*. Ed. by R. Jacquart. Vol. 156. IFIP International Federation for Information Processing. Springer US, pp. 91–120.
- Axelrod, R. (1984). *The evolution of Cooperation*. New York, NY, USA: Basic Books Inc.
- Bangemann, M. et al. (1994). “Recommendations to the European Council: Europe and the global information society”. In: *Brussels: European Commission*.
- Barber, B. (1983). *The logic and Limits of Trust*. Rutgers University Press.
- Berard, E. V. (1995). *What is a methodology*. White Paper. The Object Agency.
- Berzins, V. (2004). “Trustworthiness as risk abatement”. In: *Center for National Software Studies Workshop on Trustworthy Software*. Citeseer, p. 9.
- Bigley, G. A. and J. L. Pearce (1998). “Straining for Shared Meaning in Organization Science: Problems of Trust and Distrust”. In: *The Academy of Management Review* 23.3, pp. 405–421.
- Bimrah, K. K. (2009). “A Framework for Modelling Trust During Information Systems Development.” PhD thesis. University of East London.
- Blaze, M., J. Feigenbaum, and J. Lacy (1996). “Decentralized trust management”. In: *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*, pp. 164–173.
- Blaze, M., J. Feigenbaum, and A. D. Keromytis (1999). “KeyNote: Trust Management for Public-Key Infrastructures (Position Paper)”. In: *Proceedings of the 6th International Workshop on Security Protocols*. London, UK, UK: Springer-Verlag, pp. 59–63.
- Bostrom, R. P. and J. S. Heinen (1977). “MIS Problems and failures: a sociotechnical perspective part I: the cause”. In: *MIS Q.* 1.3, pp. 17–32.
- Braynov, S. (2002). “Contracting with uncertain level of trust”. In: *Computational Intelligence* 18, pp. 501–514.
- Bresciani, P. et al. (2004). “Tropos: An Agent-Oriented Software Development Methodology”. English. In: *Autonomous Agents and Multi-Agent Systems* 8.3, pp. 203–236.

- Brooks, A. and L. Scott (2001). “Constraints in CASE tools: results from curiosity driven research”. In: *Software Engineering Conference, 2001. Proceedings. 2001 Australian*. IEEE, pp. 285–293.
- Castelfranchi, C. and R. Falcone (1998). “Principles of trust for MAS: Cognitive anatomy, social importance, and quantification”. In: *Multi Agent Systems, 1998. Proceedings. International Conference on*. IEEE, pp. 72–79.
- (2000). “Trust Is Much More than Subjective Probability: Mental Components and Sources of Trust”. In: *Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 6 - Volume 6*. HICSS '00. Washington, DC, USA: IEEE Computer Society.
- Charney, S. (2012). “Trustworthy Computing Next”. In: *Microsoft, white paper*.
- Chu, Y. H. et al. (June 1997). “Referee: Trust Management for Web Applications”. In: *World Wide Web J.* 2.3, pp. 127–139.
- Chung, L. and J. C. S. do Prado Leite (2009). “On non-functional requirements in software engineering”. In: *Conceptual modeling: Foundations and applications*. Springer, pp. 363–379.
- Chung, L. et al. (2000). *Non-functional requirements in Software Engineering*. Kluwer Academic Publishers.
- Cofta, P. (2007). *Trust, complexity and control: confidence in a convergent world*. John Wiley.
- Cofta, P. (2008). “Towards a better citizen identification system”. English. In: *Identity in the Information Society* 1.1, pp. 39–53.
- Cofta, P., H. Laco  e, and P. Hodgson (2010). “Incorporating Social Trust into Design Practices for Secure Systems”. In: *IJDTIS* 1.4, pp. 1–24.
- Dardenne, A., A. van Lamsweerde, and S. Fickas (Apr. 1993). “Goal-directed requirements acquisition”. In: *Science of Computer Programming* 20.1-2, pp. 3–50.
- Das, T. K. and B.-S. Teng (1998). “Between trust and control: developing confidence in partner cooperation in alliances”. In: *Academy of management review* 23.3, pp. 491–512.
- Dasgupta, P. (2000). “Trust as a commodity”. In: *Trust: Making and breaking cooperative relations* 4, pp. 49–72.
- Deutsch, M. (1962). “Cooperation and trust: Some theoretical notes”. In: *Nebraska Symposium on Motivation*.
- Doherty, N. and M. King (1998). “The importance of organisational issues in systems development”. In: *Information Technology & people* 11.2, pp. 104–123.
- Drenth, P. J. (2012). *A European code of conduct for research integrity*.
- Easterbrook, S. et al. (2008). “Selecting Empirical Methods for Software Engineering Research”. In: *Guide to Advanced Empirical Software Engineering*. Ed. by F. Shull, J. Singer, and D. Sjøberg. Springer London, pp. 285–311.

- Egger, F. N. (2003). “From interactions to transactions: designing the trust experience for business-to-consumer electronic commerce”. PhD thesis.
- Elahi, G. and E. Yu (2009). “Trust trade-off analysis for security requirements engineering”. In: *Requirements Engineering Conference, 2009. RE’09. 17th IEEE International*. IEEE, pp. 243–248.
- Finnigan, D., R. Kemp, and D. Mehandjiska (2000). “Towards an ideal CASE tool”. In: *Software Methods and Tools, 2000. SMT 2000. Proceedings. International Conference on*. IEEE, pp. 189–197.
- Flyvbjerg, B. (2006). “Five Misunderstandings About Case-Study Research”. In: *Qualitative Inquiry* 12.2, pp. 219–245.
- Fortune, J. and G. Peters (2005). *Information Systems: Achieving Success by Avoiding Failure*. John Wiley & Sons.
- Friedman, B., P. H. Khan Jr, and D. C. Howe (2000). “Trust online”. In: *Communications of the ACM* 43.12, pp. 34–40.
- Gambetta, D. (1988). *Trust: Making and Breaking Cooperative Relations*. Blackwell.
- García-Magariño, I. and J. J. Gómez-Sanz (2008). “Framework for defining model language metamodels for CASE tools”. In: *Model-based Methodologies for Pervasive and Embedded Software, 2008. MOMPES 2008. 5th International Workshop on*. IEEE, pp. 14–23.
- Giorgini, P. et al. (2005). “Modeling security requirements through ownership, permission and delegation”. In: *Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on*, pp. 167–176.
- Giorgini, P. et al. (2003). “Reasoning with goal models”. In: *Conceptual Modeling—ER 2002*. Springer, pp. 167–181.
- Giorgini, P. et al. (2004). “Requirements Engineering Meets Trust Management”. In: *Trust Management*. Ed. by C. Jensen, S. Poslad, and T. Dimitrakos. Vol. 2995. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 176–190.
- Górski, J. et al. (2005). “Trust case: Justifying trust in an IT solution”. In: *Reliability Engineering & System Safety* 89.1, pp. 33–47.
- Grandison, T. (2003). “Trust Management for Internet Applications”. PhD thesis. Imperial College London.
- Haley, B. et al. (Feb. 2006). “Using Trust Assumptions with Security Requirements”. In: *Requir. Eng.* 11.2, pp. 138–151.
- Haley, C. et al. (2008). “Security Requirements Engineering: A Framework for Representation and Analysis”. In: *Software Engineering, IEEE Transactions on* 34.1, pp. 133–153.
- Hasselbring, W. and R. Reussner (2006). “Toward trustworthy software systems”. In: *Computer* 39.4, pp. 91–92.

- 
- Hatebur, D., M. Heisel, and H. Schmidt (2007). “A Pattern System for Security Requirements Engineering”. In: *Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on*. IEEE, pp. 356–365.
- Herzberg, A. et al. (2000). “Access control meets public key infrastructure, or: assigning roles to strangers”. In: *Security and Privacy, 2000. S P 2000. Proceedings. 2000 IEEE Symposium on*, pp. 2–14.
- Hoffman, L. J., K. Lawson-Jenkins, and J. Blum (July 2006). “Trust Beyond Security: An Expanded Trust Model”. In: *Commun. ACM* 49.7, pp. 94–101.
- IEEE (1990). “IEEE Standard Glossary of Software Engineering Terminology”. In: *IEEE Std 610.12*, pp. 1–84.
- Iivari, J. and R. Hirschheim (1996). “Analyzing information systems development: A comparison and analysis of eight is development approaches”. In: *Information Systems* 21.7, pp. 551–575.
- Islam, S., H. Mouratidis, and S. Wagner (2010). “Towards a framework to elicit and manage security and privacy requirements from laws and regulations”. In: *Requirements Engineering: Foundation for Software Quality*. Springer, pp. 255–261.
- Jackson, M. (1997). “The meaning of requirements”. English. In: *Annals of Software Engineering* 3.1, pp. 5–21.
- (2001). *Problem frames: analysing and structuring software development problems*. Addison-Wesley.
- Jayaswal, B. K. and P. C. Patton (2006). *Design for Trustworthy Software: Tools, Techniques, and Methodology of Developing Robust Software*. Upper Saddle River, NJ, USA: Prentice Hall PTR.
- Jøsang, A., R. Ismail, and C. Boyd (2007). “A survey of trust and reputation systems for online service provision”. In: *Decision support systems* 43.2, pp. 618–644.
- Jøsang, A., C. Keser, and T. Dimitrakos (2005). “Can We Manage Trust?” In: *Trust Management*. Ed. by P. Herrmann, V. Issarny, and S. Shiu. Vol. 3477. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 93–107.
- Jürjens, J. (2005). *Secure systems development with UML*. Vol. 1. Springer.
- Kalloniatis, C., E. Kavakli, and S. Gritzalis (2008). “Addressing privacy requirements in system design: the PriS method”. In: *Requirements Engineering* 13.3, pp. 241–255.
- Karagiannis, D. and H. Kühn (2002). “Metamodelling platforms”. In: *EC-Web*, p. 182.
- Kini, A. and J. Choobineh (1998). “Trust in electronic commerce: definition and theoretical considerations”. In: *System Sciences, 1998., Proceedings of the Thirty-First Hawaii International Conference on*. Vol. 4, 51–61 vol.4.
- Kitchenham, B. (1996). *DESMET: A method for evaluating Software Engineering methods and tools*. Tech. rep. Department of Computer Science Department of Computer Science, University of Keele.

- Kitchenham, B., S. Linkman, and D. Law (1997). “DESMET: a methodology for evaluating software engineering methods and tools”. In: *Computing Control Engineering Journal* 8.3, pp. 120–126.
- Kitchenham, B. et al. (2002). “Preliminary guidelines for empirical research in software engineering”. In: *Software Engineering, IEEE Transactions on* 28.8, pp. 721–734.
- Klopper, R., S. Gruner, and D. G. Kourie (2007). “Assessment of a framework to compare software development methodologies”. In: *Proceedings of the 2007 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries*. ACM, pp. 56–65.
- Lamsweerde, A. van (2001). “Goal-oriented requirements engineering: a guided tour”. In: *Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on*, pp. 249–262.
- Lamsweerde, A. and E. Letier (2004). “From Object Orientation to Goal Orientation: A Paradigm Shift for Requirements Engineering”. In: *Radical Innovations of Software and Systems Engineering in the Future*. Ed. by M. Wirsing, A. Knapp, and S. Balsamo. Vol. 2941. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 325–340.
- Lapouchnian, A. (2005). “Goal-Oriented Requirements Engineering - An Overview of the Current Research”. MA thesis. University of Toronto.
- Letier, E. and A. van Lamsweerde (2002). “Agent-based Tactics for Goal-oriented Requirements Elaboration”. In: *Proceedings of the 24th International Conference on Software Engineering. ICSE '02*. Orlando, Florida: ACM, pp. 83–93.
- (2004). “Reasoning about partial goal satisfaction for requirements and design engineering”. In: *Proceedings of the 12th ACM SIGSOFT twelfth international symposium on Foundations of software engineering. SIGSOFT '04/FSE-12*. Newport Beach, CA, USA: ACM, pp. 53–62.
- Lewicki, R. J. and B. B. Bunker (1996). “Developing and maintaining trust in work relationships”. In: *Trust in organizations Frontiers of theory and research*. Ed. by R. M. Kramer and T. R. Tyler. Vol. 1. Sage, pp. 114–139.
- Lewis, J. D. and A. Weigert (1985). “Trust as a social reality”. In: *Social forces* 63.4, pp. 967–985.
- Lipner, S. (2004). “The trustworthy computing security development lifecycle”. In: *Computer Security Applications Conference, 2004. 20th Annual*. IEEE, pp. 2–13.
- Luhmann, N. (1979). *Trust and Power*. Wiley.
- (1988). “Familiarity, Confidence, Trust: Problems and Alternatives”. In: *D. Gambetta, editor, Trust: Making and Breaking of Cooperative Relations, Basil Blackwell, Oxford, 1988*.



- Luna-Reyes, L. et al. (2005). “Information systems development as emergent socio-technical change: a practice approach”. In: *European Journal of Information Systems* 14.1, pp. 93–105.
- Lund, M. S., B. Solhaug, and K. Stølen (2010). *Model-driven risk analysis: the CORAS approach*. Springer.
- Lyytinen, K. and M. Newman (2008). “Explaining Information Systems Change: A Punctuated Socio-technical Change Model”. In: *European Journal of Information Systems* 17.6, pp. 589–613.
- Marcos, E. (2005). “Software engineering research versus software development”. In: *ACM SIGSOFT Software Engineering Notes* 30.4, pp. 1–7.
- Marsh, S. and P. Briggs (2009). “Examining Trust, Forgiveness and Regret as Computational Concepts”. In: *Computing with social trust*. Springer, pp. 9–43.
- Marsh, S. P. (1994). “Formalising trust as a computational concept”. PhD thesis.
- Masthoff, J. (2007). “Computationally modelling trust: an exploration”. In: *Proceedings of the SociUM workshop associated with the User Modeling conference*.
- Mayer, R. C., J. H. Davis, and F. D. Schoorman (1995). “An integrative model of organizational trust”. In: *Academy of management review* 20.3, pp. 709–734.
- McAllister, D. J. (1995). “Affect and Cognition Based Trust as Foundations for Interpersonal Cooperation in Organisations”. In: *Academy of Management Journal* 38.1, pp. 24–59.
- McGraw, D. et al. (2009). “Privacy as an enabler, not an impediment: building trust into health information exchange”. In: *Health Affairs* 28.2, pp. 416–427.
- McKnight, D. H. and N. L. Chervany (1996). *The Meanings of Trust*. Tech. rep. University of Minnesota.
- McKnight, D. H. and N. L. Chervany (2000). “What is trust? A conceptual analysis and an interdisciplinary model”. In: *AMCIS*.
- McKnight, D. H., L. L. Cummings, and N. L. Chervany (1998). “Initial trust formation in new organizational relationships”. In: *Academy of management review* 23.3, pp. 473–490.
- McKnight, D. and L. Chervany (2006). “Handbook of Trust Research”. In: ed. by R. Bachmann and A. Zaheer. Edward Elgar Publishing Ltd. Chap. Reflections on an Initial Trust-Building Model, pp. 29–51.
- Michie, D. (1982). “The state of the art in machine learning”. In: *Introductory readings in expert systems*, pp. 208–229.
- Miller, K. and J. Voas (2009). “The Metaphysics of Software Trust”. In: *IT Professional* 11.2, pp. 52–55.
- Möllering, G. (2005). “The Trust/Control Duality An Integrative Perspective on Positive Expectations of Others”. In: *International sociology* 20.3, pp. 283–305.

- Moody, D. et al. (2003). “Evaluating the quality of information models: empirical testing of a conceptual model quality framework”. In: *Software Engineering, 2003. Proceedings. 25th International Conference on*, pp. 295–305.
- Mouratidis, H. and P. Cofta (Dec. 2010). “Practitioner’s challenges in designing trust into online systems”. In: *J. Theor. Appl. Electron. Commer. Res.* 5.3, pp. 65–77.
- Mouratidis, H. and P. Giorgini (2007). “Secure Tropos: A Security-Oriented Extension of the Tropos Methodology”. In: *International Journal of Software Engineering and Knowledge Engineering* 17.02, pp. 285–309.
- Mylopoulos, J., L. Chung, and E. Yu (Jan. 1999). “From Object-Oriented to Goal-Oriented Requirements Analysis”. In: *Communications of ACM* 42.1, pp. 31–37.
- NHS. *NHS Careers*. URL: <http://www.nhscareers.nhs.uk/>.
- *NHS England*. URL: <http://www.england.nhs.uk/>.
- (2011a). *Summary Care Record Scope*. Tech. rep.
- (2011b). *The Care Record Guarantee*. Tech. rep.
- (2012a). *Introduction to the Summary Care Record*. Tech. rep.
- (2012b). *NHS Summary Care Record - Guide for GP Practice Staff.pdf*. Tech. rep.
- (2013a). *aaaaaHealth and social care information centre*. URL: <http://www.hscic.gov.uk/home>.
- (2013b). *The Handbook to the NHS Constitution*. Tech. rep.
- Nielsen, J. (1994). *Usability engineering*. Elsevier.
- Office, N. A. (2011). *The National Programme for IT in the NHS - an update on the delivery of detailed care records systems*. Tech. rep. Department of Health, UK.
- (OMG), O. M. G. *SPEM 2.0*.
- Oxford Dictionary*. URL: <http://www.oed.com/>.
- Parnas, D. L. and J. Madey (1995). “Functional documents for computer systems”. In: *Science of Computer Programming* 25.1, pp. 41–61.
- Partridge, D. (1997). “The case for inductive programming”. In: *Computer* 30.1, pp. 36–41.
- Pavlidis, M. et al. (2012). “Dealing with trust and control: A meta-model for trustworthy information systems development”. In: *Research Challenges in Information Science (RCIS), 2012 Sixth International Conference on*, pp. 1–9.
- Pavlidis, M., H. Mouratidis, and S. Islam (2012). “Modelling security using trust based concepts”. In: *International Journal of Secure Software Engineering (IJSSE)* 3.2, pp. 36–53.
- Pavlidis, M. et al. (2014). “Modeling Trust Relationships for Developing Trustworthy Information Systems”. In: *International Journal of Information System Modeling and Design (IJISMD)* 5.1, pp. 25–48.
- Pearson, S. and B. Balacheff (2003). *Trusted computing platforms: TCPA technology in context*. Prentice Hall Professional.

- Phillips, C. et al. (1998). “The usability component of a framework for the evaluation of OO CASE tools”. In: *Software Engineering: Education & Practice, 1998. Proceedings. 1998 International Conference*. IEEE, pp. 134–141.
- Pourshahid, A. and T. Tran (2007). “Modeling trust in e-commerce: an approach based on user requirements”. In: *Proceedings of the ninth international conference on Electronic commerce*. ACM, pp. 413–422.
- Presti, S. L. et al. (2006). “Holistic Trust Design of E-Services”. In: *Trust in E-Services: Technologies, Practices and Challenges*, pp. 113–139.
- Public Accounts, C. of (2013). *The dismantled National Programme for IT in the NHS*. Tech. rep. House of Commons.
- Rasmusson, L. and S. Jansson (1996). “Simulated social control for secure Internet commerce”. In: *Proceedings of the 1996 workshop on New security paradigms*. ACM, pp. 18–25.
- Ray, I. and S. Chakraborty (2004). “A Vector Model of Trust for Developing Trustworthy Systems”. In: *Computer Security – ESORICS 2004*. Ed. by P. Samarati et al. Vol. 3193. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 260–275.
- Rempel, J. K., J. G. Holmes, and M. P. Zanna (1985). “Trust in close relationships”. In: *Journal of personality and social psychology* 49.1, p. 95.
- Riegelsberger, J., M. A. Sasse, and J. D. McCarthy (2005). “The mechanics of trust: A framework for research and design”. In: *International Journal of Human-Computer Studies* 62.3, pp. 381–422.
- Rohm, A. J. and G. R. Milne (2004). “Just what the doctor ordered: the role of information sensitivity and trust in reducing medical information privacy concern”. In: *Journal of Business Research* 57.9, pp. 1000–1011.
- Ross, D. and J. Schoman K.E. (1977). “Structured Analysis for Requirements Definition”. In: *Software Engineering, IEEE Transactions on SE-3.1*, pp. 6–15.
- Rousseau, D. M. et al. (1998). “Not so different after all: A cross-discipline view of trust”. In: *Academy of management review* 23.3, pp. 393–404.
- Roy, M. C., O. Dewit, and B. A. Aubert (2001). “The impact of interface usability on trust in web retailers”. In: *Internet research* 11.5, pp. 388–398.
- Runeson, P. and M. Höst (2009). “Guidelines for conducting and reporting case study research in software engineering”. English. In: *Empirical Software Engineering* 14.2, pp. 131–164.
- Sabater, J. and C. Sierra (2005). “Review on computational trust and reputation models”. In: *Artificial Intelligence Review* 24.1, pp. 33–60.
- Sasse, M. A. (2005). “Usability and trust in information systems”. In: *Trust and Crime in Information Societies*.
- Schillo, M., P. Funk, and M. Rovatsos (2000). “Using trust for detecting deceitful agents in artificial societies”. In: *Applied Artificial Intelligence* 14.8, pp. 825–848.

- Schneider, F., S. Bellovin, and A. Inouye (1999). “Building trustworthy systems: lessons from the PTN and Internet”. In: *Internet Computing, IEEE* 3.6, pp. 64–72.
- Seaman, C. (1999). “Qualitative methods in empirical studies of software engineering”. In: *Software Engineering, IEEE Transactions on* 25.4, pp. 557–572. ISSN: 0098-5589. DOI: 10.1109/32.799955.
- Seffah, A. and J. Rilling (2001). “Investigating the relationship between usability and conceptual gaps for human-centric CASE tools”. In: *Human-Centric Computing Languages and Environments, 2001. Proceedings IEEE Symposia on*. IEEE, pp. 226–231.
- Shackel, B. (1991). “Usability-context, framework, definition, design and evaluation”. In: *Human factors for informatics usability*, pp. 21–37.
- Shaw, M. (2003). “Writing good software engineering research papers: minitutorial”. In: *Proceedings of the 25th International Conference on Software Engineering*. ICSE '03. Portland, Oregon: IEEE Computer Society, pp. 726–736.
- Shneiderman, B. (Dec. 2000). “Designing Trust into Online Experiences”. In: *Commun. ACM* 43.12, pp. 57–59. ISSN: 0001-0782. DOI: 10.1145/355112.355124. URL: <http://doi.acm.org/10.1145/355112.355124>.
- Sjoeberg, D. et al. (2005). “A survey of controlled experiments in software engineering”. In: *Software Engineering, IEEE Transactions on* 31.9, pp. 733–753. ISSN: 0098-5589. DOI: 10.1109/TSE.2005.97.
- Society, B. C. *BCS - The Chartered Institute for IT*.
- Stapelberg, R. F. (2009). *Handbook of reliability, availability, maintainability and safety in engineering design*. Springer.
- Sutcliffe, A. (2006). “Trust: From Cognition to Conceptual Models and Design”. In: *Advanced Information Systems Engineering*. Ed. by E. Dubois and K. Pohl. Vol. 4001. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 3–17.
- Tan, Y.-H. (2003). “A Trust Matrix Model for Electronic Commerce”. In: *Proceedings of the 1st International Conference on Trust Management*. iTrust'03. Heraklion, Crete, Greece: Springer-Verlag, pp. 33–45.
- Uddin, M. G. and M. Zulkernine (2008). “UMLtrust: towards developing trust-aware software”. In: *Proceedings of the 2008 ACM symposium on Applied computing*. SAC '08. Fortaleza, Ceara, Brazil: ACM, pp. 831–836.
- Van Lamsweerde, A. (2000). “Requirements engineering in the year 00: a research perspective”. In: *Proceedings of the 2000 International Conference on Software Engineering*, pp. 5–19.
- Van Lamsweerde, A., R. Darimont, and E. Letier (1998). “Managing conflicts in goal-driven requirements engineering”. In: *IEEE Transactions on Software Engineering* 24.11, pp. 908–926.

- Van Lamsweerde, A. and L. Willemet (1998). “Inferring declarative requirements specifications from operational scenarios”. In: *IEEE Transactions on Software Engineering* 24.12, pp. 1089–1114.
- Van Lamsweerde, A. and E. Letier (2000). “Handling Obstacles in Goal-Oriented Requirements Engineering”. In: *Software Engineering, IEEE Transactions on* 26.10, pp. 978–1005.
- Van Lamsweerde, A. et al. (2007). “Engineering requirements for system reliability and security”. In: *NATO Security through Science Series D - Information and Communication Security* 9, p. 196.
- Verner, J. et al. (2009). “Guidelines for industrially-based multiple case studies in software engineering”. In: *Research Challenges in Information Science, 2009. RCIS 2009. Third International Conference on*, pp. 313–324.
- Viega, J., T. Kohno, and B. Potter (Feb. 2001). “Trust (and Mistrust) in Secure Applications”. In: *Commun. ACM* 44.2, pp. 31–36.
- Wieringa, R. and J. Heerkens (2006). “The methodological soundness of requirements engineering papers: a conceptual framework and two case studies”. In: *Requirements Engineering* 11.4, pp. 295–307.
- Wieringa, R. and A. Morali (2012). “Technical Action Research as a Validation Method in Information Systems Design Science”. In: *Design Science Research in Information Systems. Advances in Theory and Practice*. Ed. by K. Peffers, M. Rothenberger, and B. Kuechler. Vol. 7286. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 220–238.
- Wilson, M. et al. (2007). “The TrustCoM approach to enforcing agreements between interoperating enterprises”. In: *Enterprise Interoperability*. Springer, pp. 365–375.
- Wohlin, C. et al. (2012). *Experimentation in software engineering*. Springer (first edition by Kluwer in 2000).
- Yan, Z. and P. Cofta (2003). “Methodology to Bridge Different Domains of Trust in Mobile Communications”. English. In: *Trust Management*. Ed. by P. Nixon and S. Terzis. Vol. 2692. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 211–224.
- Yu, E. (1993). “Modeling organizations for information systems requirements engineering”. In: *Proceedings of IEEE International Symposium on Requirements Engineering*, pp. 34–41. DOI: 10.1109/ISRE.1993.324839.
- Yu, E. (1995). “Modelling Strategic Relationships for Process Reengineering”. PhD thesis. University of Toronto.
- Yu, E. and L. Liu (2001). “Modelling Trust for System Design Using the i \* Strategic Actors Framework”. English. In: *Trust in Cyber-societies*. Ed. by R. Falcone, M. Singh, and Y.-H. Tan. Vol. 2246. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 175–194.

- Yu, E. et al. (2011). *Social Modeling for Requirements Engineering*. The MIT Press.
- Yue, K (1987). “What Does It Mean to Say that a Specification is Complete?” In: *Fourth International Workshop on Software Specification and Design (IWSSD-4)*, Monterey, USA.
- Zarrabi, F. et al. (2012). “A Meta-model for Legal Compliance and Trustworthiness of Information Systems”. In: *Advanced Information Systems Engineering Workshops*. Ed. by M. Bajec and J. Eder. Vol. 112. Lecture Notes in Business Information Processing. Springer Berlin Heidelberg, pp. 46–60.
- Zave, P. (1997). “Classification of research efforts in requirements engineering”. In: *ACM Computing Surveys (CSUR)* 29.4, pp. 315–321.
- Zelkowitz, M. V. and D. Wallace (1997). “Experimental Validation In Software Engineering”. In: *Information and Software Technology* 39, pp. 735–743.
- Zelkowitz, M. and D. Wallace (1998). “Experimental models for validating technology”. In: *Computer* 31.5, pp. 23–31. ISSN: 0018-9162. DOI: 10.1109/2.675630.