Department of Computer Science
Series of Publications A
Report A-2012-4

The effect of reputation on trust decisions in inter-enterprise collaborations

Sini Ruohomaa

To be presented, with the permission of the Faculty of Science of the University of Helsinki, for public criticism in Auditorium XIV, University of Helsinki Main Building, on 4 May 2012 at noon.

University of Helsinki
Finland

Supervisor
Lea Kutvonen

Pre-examiners
Ehud Gudes
Patrick Hung

Opponent
Audun Jøsang

Custos
Jussi Kangasharju

Contact information

Department of Computer Science
P.O. Box 68 (Gustaf Hällströmin katu 2b)
FI-00014 University of Helsinki
Finland

Email address: postmaster@cs.helsinki.fi
URL: http://www.cs.Helsinki.fi/
Telephone: +358 9 1911, telefax: +358 9 191 51120

Copyright © 2012 Sini Ruohomaa
ISSN 1238-8645
ISBN 978-952-10-7911-5 (paperback)
ISBN 978-952-10-7912-2 (PDF)
Computing Reviews (1998) Classification: C.2.4, H.5.3, K.4.4, K.6.5
Helsinki 2012
Unigrafia

The effect of reputation on trust decisions in inter-enterprise collaborations

Sini Ruohomaa

Department of Computer Science
P.O. Box 68, FI-00014 University of Helsinki, Finland
sini.ruohomaa@cs.helsinki.fi, http://cinco.cs.helsinki.fi/

PhD Thesis, Series of Publications A, Report A-2012-4
Helsinki, April 2012, x + 214 + 44 pages
ISSN 1238-8645
ISBN 978-952-10-7911-5 (paperback)
ISBN 978-952-10-7912-2 (PDF)

Abstract

Inter-enterprise collaboration has become essential for the success of enterprises. Competition between supply chains and networks of enterprises brings with it a need to participate in multiple collaborations simultaneously. We propose to increase the automation of the involved routine processes where possible, which in turn reduces the cost and effort of collaboration establishment and management.

All inter-enterprise collaboration builds on trust. Relying on autonomous partner enterprises for a joint venture creates uncertainty and risks, which must be balanced by a willingness to collaborate despite the loss of control. Trust decisions are first made on establishing the collaboration, to ensure an acceptable balance between the risks of the commitment and its benefits to the participant. In addition, measures are taken during the collaboration in order to keep this balance.

We define a trust management architecture where autonomous actors make automated, local trust decisions based on private policy. The decision-making process considers identified risks to assets as well as incentives, and learns from both first-hand and globally shared experience on the behaviour of business peers. While trust decisions help protect enterprise assets locally, a feedback loop through shared experiences also provides social pressure to respect contracts in the entire ecosystem. Trust decisions can be adjusted to different and changing business situations through policy configurations. Policy-defined routine decisions are automated, while other cases are passed to a human user. The trust management system is a part of the Pilarcos collaboration management middleware, which provides collaboration management services, including partner discovery and interoperability checking, contract negotiation, breach detection and recovery.

The trust management architecture encompasses four contributions: First, we define an information model for multidimensional trust decisions. Second, we define the algorithm and process for making trust decisions privately at specific points of the collaboration, based on a combination of private and shared information. Third, we define aggregation algorithms for reputation, which enables the system to learn from past experiences that are shared between the ecosystem members through reputation systems. Fourth, we specify methods for adjusting trust decisions to different and changing business situations. To evaluate the architecture, we have implemented it as the TuBE (Trust Based on Evidence) trust management system, and analyze its properties, such as adjustability to different business situations and attack resistance.

The results of this work can be utilized by the service provider enterprises, through increased computational support for collaboration management, by operators in designing the infrastructure services, and to standardize best practices and create new collaboration patterns within business domains.

Computing Reviews (1998) Categories and Subject Descriptors:

C.2.4	Computer-Communication Networks: Distributed Systems
H.5.3	Information Interfaces and Presentation: Group and Organisation Interfaces
K.4.4	Management of Computing and Information Systems: Electronic Commerce
K.6.5	Management of Computing and Information Systems: Security and Protection

General Terms:
trust management, trust decisions, reputation, inter-enterprise collaboration

Additional Key Words and Phrases:
enterprise assets in open service ecosystems, risk, reputation credibility

Acknowledgements

I have learned from my colleague Laila Daniel that a good acknowledgements section utterly demolishes the illusion of a PhD thesis being a solo effort. Most notably, it is thanks to a whole society of people *not* pursuing research careers that some more or less kooky individuals in it, such as myself, can dedicate their efforts to years of searching for something that may or may not be a pot of gold at the end of the rainbow. Or, as a wise person once put it, going out to explore the thin ice because it is not so dangerous if we are the ones to fall in. Risk-taking, in terms of potential for unexpected or non-profitable results, is an integral part of research.

All in all, if we have seen a little further, it is by being hauled on top a veritable human pyramid of joint effort. Instead of giants, I just see people and more people.

Of my fellow searchers, I first thank my supervisor Lea Kutvonen, who has been the primary factor in turning my cost-benefit assessment of an academic career towards the positive, despite my access to numerous third-party experience reports indicating that research is not a rational choice of profession. Whatever my scientific career ends up amounting to, it will be because Lea helped me retain my sanity the first years while I was urgently growing a confidence as a researcher. In addition, she is the kind of visionary who finds the most splendid questions to ask, ones that endure extensive poking and prodding over years and years from different angles, but towards a single goal that we can all agree on in the group. Good questions make the rest of the research effort infinitely more rewarding.

I want to praise the present and past members of our Collaborative and Interoperable Computing research group (CINCO), who are a set of experts on quite different areas brought together by a common goal. It is extremely satisfying to learn new things through working with people holding so many different flavours of knowledge. I have worked most closely with Lea Viljanen, who started the dig for trust management solutions with me by applying a security perspective to the problem, and Janne Metso, who extended my point of view to the Pilarcos system and with whom I have had many fascinating discussions about monitoring and the handling of collaboration contracts. I wish to also thank Puneet Kaur, who joined us last year, for sharing my fascination with trust management and helping me envision a next generation of researchers I might support on their way.

Of my mentors, I particularly wish to thank Tiina Niklander, the specialization line go-to person and the instructor who tipped me in the right direction back when I was writing my Bachelor thesis. I also thank Jussi Kangasharju and Kaisa Nyberg for their support through the Helsinki Doctoral Programme in Computer Science, HeCSE, and Patrik Floréen for his good questions.

I thank my pre-examiners, Ehud Gudes and Patrik K. Hung, for their encouraging feedback and helpful comments. I also wish to thank Audun Jøsang, my assigned opponent, whose lifetime contributions in the field of computational trust and reputation management have influenced me to the extent of even making me study things that I did not really want to know about.

I have had the pleasure of meeting and exchanging thoughts with many extraordinary people on the way, too many to exhaustively list. Nearly every conference trip has given me a sleuth of new ideas and connections with brilliant people, particularly in the iTrust / IFIP Trust Management conference series.

Back home, the students and staff at the Department of Computer Science make it an invigorating place to work in. The energy for renewal generated by more than a hundred new faces coming in to the department every year is mind-blowing, and the community built by the active student organization TKO-äly allows us all to benefit from it.

I extend special thanks to my mother, whose support with the primary things in personal life has allowed me to pursue secondary matters, like career advancement, with a peace of mind. I can only wish that every working parent had sponsors like mine. I also thank my husband, Mikko, both for improving the world for us both while I am busy elsewhere, and for thoroughly *grokking* me. My son has contributed some colour to this book by doodling pictures of slow loris and their worm-rich staple on the drafts; alas, they did not make it to the final version.

I wish to toast this book to unusual ways of thinking.

Helsinki, 2 April 2012
Sini Ruohomaa

Contents

Chapter 1

Introduction

This dissertation proposes a trust management system for inter-enterprise collaborations in the open service market. In this chapter, we motivate the work, describe the problem statement, summarize our main contributions and review the research history behind the dissertation.

1.1 Motivation

The importance of inter-enterprise collaboration has been strongly increasing in the past years, and the need for supporting solutions continues to rise. Enterprises enter into collaborations providing mutual benefit, and seek a minimum of complication and overhead cost in setting up and managing these shared projects.

The current trend is for enterprises to focus on their core competence, aiming for a competitive edge through excelling in a few strategically chosen functions rather than spreading resources too thinly over a broad market. On the other hand, composite services are still needed: end users and customers are attracted by the convenience of e.g. holiday packages that include transportation, accommodation and value-added activities on the location. Process chains, such as the production of paper from wood, are also a typical field of collaboration between multiple specialists sharing a goal.

In the human-driven, centralized approach[1], a collaboration would be formed through one dominant actor building a set of partners for itself over

[1]Represented in e.g. the FP7 work programme [Eur07], roadmaps for Enterprise Interoperability [LCDP06], Service-Oriented Computing [PTD$^+$06, Huh06] and digital business ecosystems [NDA$^+$07] research, as well as specific projects such as NESSI [FAB$^+$06], Athena [The05] and CrossWork [MG10].

the course of years. The collaboration would then form around this hub actor. For example, in the case of holiday packages, a travel agency might provide one of the services itself and have subcontractors for the rest. In the forest industry, the forestry company owning the factory could centralize all operations through pairwise contracts with the forest owners, loggers and transport companies. These hub actors are difficult to replace, and all other actors must perform systems integration to be able to work with the largest actor, unless they are networked broadly enough to remain partially independent of it.

Small actors can remain independent when the service ecosystem and its supporting infrastructure is built to make setting up and operating collaborations simple, so that interoperability requires relatively small investments for each business network joined. In this setting, business networks of small to medium enterprises are free to compete against large enterprises, and gain more influence in negotiations than they could by themselves. On the other hand, large companies can organize work within themselves and with external partners more efficiently as well. All this contributes towards a more open market.

As inter-enterprise collaboration moves towards these open systems, participants remain autonomous in collaborations rather than becoming centrally controlled, and they retain their private agendas. As enterprises join and participate in multiple business networks, joining and managing the collaborations must be made efficient.

Efficient collaboration management requires a globally accessible infrastructure that provides automation support for any routine processes. These new facilities must support the setup and management of business networks in a way that reflects the collaborators' business needs, providing for discovery and selection of business partners and services, contracting, monitoring the success of a collaboration, and policy enforcement. Automating these routine processes demands that information used in them is made accessible to automation tools: it must be structured through models, and protected with policy.

However, these basic services alone are not sufficient to form a valid, scalable ecosystem of services and collaborations. As the service ecosystem grows in size, the unavoidable presence of uncooperative, misbehaving actors will quickly begin to deteriorate it, eventually making collaboration impossible. A form of social control — pressure to follow shared norms — is needed to keep the market alive [FF03, RZFK00].

All inter-enterprise collaboration builds on trust between the actors. Relying on an autonomous partner enterprise for a joint venture removes some control over the outcome, which in turn creates risks. These risks must be balanced by an expectation of a greater benefit. As the future is unknown, the expectations must build on the past: experiences on the past behaviour of business peers create a perception of their intentions and norms, which then provide a basis for predicting their future behaviour. Sharing these experiences globally through reputation systems provides the social control that traditionally has been represented by e.g. accreditors such as the Better Business Bureau [Bet10] and Trustmark [Tru10], credit ratings such as Standard and Poors [Sta10], blacklists such as those provided by the National Consumer Agency of Ireland [Nat10], stock market fluctuations and word-of-mouth. In all contractually regulated collaborations, legal recourse remains a final, costly and slow but effective option; however, social pressure allows the market to thrive with considerably less centralized control.

Trust decisions are first made on whether to join a collaboration, weighing whether the risks and benefits of the commitment are in balance from the point of view of the actor making the decision. In addition, measures must be taken during the collaboration to ensure that this balance is kept. The behaviour of a partner is continuously monitored both locally and through globally shared experiences. If malicious or incorrect behaviour is observed, or the collaboration otherwise takes a less beneficial turn, the enterprise can decide to take action to remove the misbehaving peer, or to withdraw from the collaboration itself.

All actors must build their own estimations out of incomplete information, evaluate the credibility of the information locally, and detect when there simply is not enough information to make a well-based decision. We should only aim to automate decisions in routine cases: unpredictable situations where information is insufficient or the stakes too high must be delegated to human administration to evaluate. Identifying these situations requires a measure of the quality of the available information as well as a notion of the scale of risk involved.

In order to support the development of dynamic inter-enterprise collaborations and respond to this need for introducing a computational equivalent of trust into the open service market, we propose a trust management architecture. It should have the following properties:

- an information model for trust decisions, including ways to express experiences, risks and risk acceptance, incentives and a changing context around the decision,
- algorithms and specified points for making trust decisions privately within each enterprise based on the available information, and a way

to distinguish between automatic routine decisions and cases requiring human intervention,

- aggregation algorithms for experience information, received both locally and globally through reputation systems, to ensure that the trust decisions adjust to new experience, and

- support for adjusting the decisions to different business situations.

This kind of trust management system will protect local enterprise assets from risky collaborations. Even more importantly, through its use of shared experience information, it will allow the entire service ecosystem to scale up in size through providing social control and deterring misbehaviour.

1.2 Automating trust decisions for inter-enterprise collaborations

In the open service market, collaborating service providers rely on other collaborators to fulfil their part of a contract. In this environment, traditional security measures like access control and firewalls, directed towards protecting enterprise resources from unauthorized use, are no longer sufficient [RJ96]. A degree of distrust is needed even towards partners who have been trusted with authorization.

Current approaches to automating trust decisions can be divided into two groups: policy-based systems, which validate and combine certificates of trustworthiness from a low number of central sources (e.g. PolicyMaker [BFL96], KeyNote [BFK98], REFEREE [CFL$^+$97], KAoS [UBJ04], Ponder [DDLS01] and WS-Trust [OAS07]), and purely reputation-based systems, which use for example probabilistic methods to combine trustworthiness information from a high number of less reliable or knowledgeable sources (e.g. Beta [JI02], Travos [TPJL06], Regret [SS02], EigenTrust [KSGM03] and UniTEC [KR03]).

The strength of the policy-based approach lies in being able to use the same trustworthiness information in multiple situations. In contrast, purely reputation-based systems tend to incorporate decision policy into the model for acquiring experiences, which limits the usability of the collected information.

The strength of reputation-based systems is in the efficient distribution, evaluation and updating of trustworthiness information. In policy-based systems, the updating process of certificates forming the core of trustworthiness information is left unaddressed — certificates are typically manually issued and revoked.

We propose to combine the strengths of these two approaches within a novel trust management architecture, TuBE (Trust Based on Evidence). The proposed architecture manages and uses up-to-date reputation information as a basis for risk estimation, and extends the policy-based approach to combine the information with local business rules and valuations to determine what kinds of risks are tolerable.

TuBE provides the means for autonomous actors to make automated, local trust decisions that evaluate the risks and incentives of a collaboration. Decisions are made when joining a collaboration and during its operation, when there is a need to commit additional resources. The decisions are based on private policies, using experience information that is shared through reputation systems and locally evaluated for credibility.

Our work connects the fields of inter-enterprise collaboration management, policy-based decision systems and reputation management into the folds of a single architecture, which gives the solutions from each field access to much-needed support from each other. Following this philosophy, we analyze related work from four contexts in this dissertation, with different connections to our work: inter-enterprise collaboration incorporating a basic level of trust-awareness; trust management models to support trust decision policies from the perspective of risks and incentives; reputation management models for computationally analyzing flows of reputation information and transforming it into a format to support risk analysis; and policy analysis through simulation and game theory, which aim to provide a better understanding of the effects and suitability of trust management policies to different environments and situations.

TuBE ties into all of these four domains, and must address important questions that have been studied in each of them separately, such as: Where do experiences come from? How are they converted into the format accepted by the reputation management system? Where does the trust decision system gain its risk and incentive information? Does the system only support a single decision context, and is it sufficient for that context? How widely does the reputation system need to be used in order for the experiences shared in it to be sufficient to support the decisions? How does trust management relate to other enterprise policies? The architecture also clarifies the relationship between trust, reputation and risk, which has been previously found problematic [MMH02, JP04, RK05].

The TuBE trust management system is a part of the Pilarcos collaboration management infrastructure [KMR05, KRM07], which provides services for partner discovery, interoperability checking, and contract negotiation, as well as runtime breach detection and recovery. The surrounding Pilarcos infrastructure services provide the trust management system with important input on local policies and attributes of the collaboration being considered, as well as a monitoring interface for gathering local information about the behaviour of the business peers. The Pilarcos infrastructure services provide support for technical and semantic interoperability between services, but do not fully address the question of pragmatic interoperability: whether two services in fact want to collaborate. TuBE focuses on this question from the point of view of trust. The remaining aspects of pragmatic interoperability tie into choosing collaborations that match enterprise strategy and investment, and avoidance of contracts that would contradict local policies; these are covered elsewhere in the architecture [KRM08, RK10].

1.3 Main contributions

More concretely, the contributions of this thesis are fourfold:

1. We define an information model for multi-dimensional trust decisions, which is necessary for automating routine decisions. The model offers support for dynamic trust decision policies that consider both the risks and incentives of inter-enterprise collaboration in the open service market.

2. We define the algorithms for making trust decisions privately within an enterprise, based on a combination of private and shared information. These algorithms are designed to not depend on all enterprises using the same tools, as enforcing homogeny is not plausible in open service ecosystems.

3. We define the aggregation algorithms for receiving experience information locally and through reputation systems, and for interpreting and evaluating the credibility of the shared reputation information locally. These reputation flows allow the architecture to learn from experience.

4. We specify points in the architecture where the basic behaviour of the system can be controlled through private policies and metapolicies, and study the behaviour of central example policies to illustrate the flexibility of the system, and to contribute to the understanding the

effects of different policies. Together, these policies and metapolicies allow trust decisions to adjust to different business situations.

We propose a system for automating trust-based decision-making for inter-enterprise collaborations in routine cases, and automatically identifying non-routine situations where human intervention is needed instead. It can be easily adjusted to different and changing business situations, so it can serve enterprises from different business domains, and can support trust decisions both in open service ecosystems as well as in more closed environments, such as virtual breeding environments of already-known strategic business partners.

The TuBE architecture is designed to be modular, forming a framework into which more specialized solutions can be plugged. The core trust decision algorithms are designed to be computationally simple enough that automated decision-making can be performed in real time. Should performance not be an issue, we do not limit how complex the decision policies themselves can be made. This allows plugging in more complex and mathematically powerful approaches, such as risk evaluation based on probability distributions (e.g. Beta [JI02]) or possibility theory [CW$^+$08]. Similarly, TuBE is designed to be able to take in reputation information from multiple different sources, as we consider it probable that different business domains will not converge into using a single reputation system.

TuBE is also designed to be adjustable, but not at the expense of configurability: for simple decision contexts, the system can be set up to operate on a small set of default policies, while it can be adjusted to more elaborate environments either manually, or through semi-automatically refining higher-level policies and models into TuBE rules. As a design choice, policies for integrating new information from sources of varying quality are kept separate from decision policies using the information; this ensures that the same information can be used in very different contexts of decision-making. This flexibility of the architecture ensures that TuBE can be used both by small and large enterprises for trust management in the open service ecosystem.

1.4 Research history

The research reported in this thesis has been done as a part of the CINCO research group, which has been developing the Pilarcos infrastructure services [KRRM08]. The trust management system work began in the TuBE (Trust Based on Evidence) project in 2004, which was funded by Tekes, Nixu and Stonesoft. As a part of the EU InterOP Network of Excellence, we have focused on studying interoperability issues in trust, policies and other non-functional aspects of collaboration [INT05, INT07b, INT07a]. Two workshops on Interoperability solutions to Trust, Security, Policies and QoS for Enhanced Enterprise Systems (IS-TSPQ) were organized as a part of the dissemination activities [KLMR07].

My research work can be divided into four interleaved phases: surveying the field and identifying central trust management concepts, defining the trust information model and algorithms, designing and implementing the TuBE architecture and its connections to Pilarcos, and evaluating the architecture, including simulation experiments on a research prototype of the trust management system.

My survey work has aimed to study the related research fields, to identify existing solutions and other useful results and to build a model of trust to use in the TuBE project. The central publications from this phase have been two surveys, in addition to which I have disseminated results from this phase in Finland and near areas [RK06, Ruo06]. The survey work represents an important phase in the research leading up to this thesis; these articles are thus included in the appendix to be evaluated as a part of the contribution. In contrast, the related work analysis in the following chapters has been done with a narrower scope, specifically chosen to illustrate the concepts and goals discussed in the thesis.

Ruohomaa, S. and Kutvonen, L., Trust management survey. *Proceedings of the iTrust 3rd International Conference on Trust Management*, Rocquencourt, France, volume 3477 of Lecture Notes in Computer Science. Springer-Verlag, May 2005, pages 77–92.

- The article provides a general overview of the state of the art in trust management. I studied the literature based on a framework for comparing the state of the art, and conducted the analysis.

- The iTrust conference was a key cross-disciplinary conference on trust and trust management within Europe. In 2007, it formed the basis of the IFIP Working Group 11.11 and its IFIPTM conference.

Ruohomaa, S., Kutvonen, L. and Koutrouli, E., Reputation management survey. *Proceedings of the 2nd International Conference on Availability, Reliability and Security (ARES 2007)*, Vienna, Austria, April 2007, IEEE Computer Society, pages 103–111.

- The article analyzes 11 reputation systems from the point of view of information credibility analysis. This survey was joint work with the

University of Athens; my contribution focused on the framework for analysis and expertise on a subset of the systems compared.

Sini Ruohomaa and Lea Kutvonen. Making multi-dimensional trust decisions on inter-enterprise collaborations. *Proceedings of the Third International Conference on Availability, Security and Reliability (ARES 2008)*, pages 873-880, Barcelona, Spain, March 2008. IEEE Computer Society.

- The article presents the information model and algorithms for computational trust decisions, as well as managing and updating trust information. I refined the preliminary trust model from the TuBE project into the more detailed model presented in the article.

Sini Ruohomaa, Lea Viljanen, and Lea Kutvonen. Guarding enterprise collaborations with trust decisions - the TuBE approach. *Interoperability for Enterprise Software and Applications. Proceedings of the Workshops and the Doctoral Symposium of the Second IFAC/IFIP I-ESA International Conference: EI2N, WSI, IS-TSPQ 2006*, pages 237-248. ISTE Ltd.

- The article presents the TuBE system architecture and its interfaces to the Pilarcos middleware. The basic architecture was joint work that began with the TuBE project; I refined the architecture further, and held the main responsibility in interfacing the trust management component to the Pilarcos infrastructure.
- The publication forum was the aforementioned IS-TSPQ workshop, a part of the dissemination activities of the InterOP Network of Excellence.

Lea Kutvonen, Janne Metso, and Sini Ruohomaa. From trading to eCommunity population: Responding to social and contractual challenges. *Proceedings of the 10th IEEE International EDOC Conference (EDOC 2006)*, pages 199-210, Hong Kong, October 2006. IEEE. Best paper award.

- The article presents challenges and solutions in setting up an inter-enterprise collaboration. My contribution focused on analyzing the trust management aspects of finding and selecting collaboration partners. One of the conclusions in the paper was that the population process must be based on public information when the populator is an external third-party service, in order to not force all initiator organizations to reveal more sensitive information on e.g. their partner preferences. More general trust-related requirements, such as required certification of competence for a particular task, can be pushed into the parameters given to the populator if they have been left optional in the collaboration model.

- EDOC is a key conference for inter-enterprise collaboration, with consistently low acceptance rates.

Lea Kutvonen, Janne Metso, and Sini Ruohomaa. From trading to eCommunity management: Responding to social and contractual challenges. *Information Systems Frontiers (ISF) — Special Issue on Enterprise Services Computing: Evolution and Challenges*, 9(2-3):181-194, July 2007.

- The journal article following the EDOC article shifts the focus of the work towards the new infrastructure services needed in different phases of the collaboration, as well as operational time issues in collaboration management. My contribution focused on the trust management infrastructure.

Lea Kutvonen, Toni Ruokolainen, Sini Ruohomaa, and Janne Metso. Service-oriented middleware for managing inter-enterprise collaborations. *Global Implications of Modern Enterprise Information Systems: Technologies and Applications, Advances in Enterprise Information Systems (AEIS)*, pages 209-241. IGI Global, December 2008.

- The book chapter analyzes the operating environment in a broader perspective, providing an overview of the entire existing Pilarcos infrastructure, including its trust management services. My contribution involved the presentation of the trust management infrastructure.

Lea Kutvonen, Sini Ruohomaa, and Janne Metso. Automating decisions for inter-enterprise collaboration management. *Pervasive Collaborative Networks. IFIP TC 5 WG 5.5 Ninth Working Conference on Virtual Enterprises*, pages 127-134, Poznan, Poland, September 2008. Springer.

- The article focuses on metapolicies needed to support decision-making on three different levels: first, whether a collaboration is relevant in relation to the business strategy of the enterprise, second, whether it is valid, i.e. in accordance with for example the local privacy policy and legislation, and third, whether it is worth the risk. My contribution covered trust decisions, which appear on the third level; the decision-making environment influences the requirements for the trust management system as well as the decision context information that can be expected to be available.

Sini Ruohomaa, Lea Kutvonen. Trust and distrust in adaptive inter-enterprise collaboration management. *Journal of Theoretical and Applied Electronic Commerce Research — Special Issue on Trust and Trust Management*, August 2010.

- The article analyzes the role of different policies in trust management, and provides an overview of the information model and algorithms for making trust decisions that can be adjusted to different and changing business situations. It also evaluates the cost of implementing the proposed trust management as well as the broader policy-driven decision-making system. My contribution covers the trust management focus point of the article.

Sini Ruohomaa, Aleksi Hankalahti, Lea Kutvonen. Detecting and reacting to changes in reputation flows. *Proceedings of the Fifth IFIP WG 11.11 International Conference on Trust Management*, June 2011.

- The article introduces reputation epochs, which are used to capture changes in behaviour. It evaluates the cost of implementing them and presents simulations of example epoch change policies to demonstrate their effects on decision-making. Reputation epochs are my original concept.

- A key conference in trust management, the IFIPTM conference continues the aforementioned cross-disciplinary iTrust conference series.

Sini Ruohomaa, Puneet Kaur, Lea Kutvonen. From subjective reputation to verifiable experiences — augmenting peer-control mechanisms for open service ecosystems. *Proceedings of the Sixth IFIP WG 11.11 International Conference on Trust Management*, May 2012. To appear.

- The article introduces an objective form of reputation, where experiences are based on contract-defined outcomes and verified by third party witnesses. It evaluates the costs and benefits of such a scheme. Separate aspects of the approach can be found in related work, e.g. the idea of defining objective experiences in contracts is comparable to sharing evaluation information based on industry-adopted key indicators, while this combination is my original concept.

- The IFIPTM conference is a key conference in trust management.

1.5 Thesis structure

The rest of the thesis is organized as follows:

Chapter 2 sets the trust management architecture in its application area of inter-enterprise collaboration. It defines the central concepts for, and specifies the requirements of trust management in open business networks, formulating the research problem. The chapter also sets the goals of the architecture, which form the basis for evaluating the proposed system. Related work on trust-aware inter-enterprise collaboration management is presented in this chapter, in order to compare the goals and chosen points of focus for existing similar proposals to those in Pilarcos and TuBE.

Chapter 3 focuses on automated trust management. It defines the trust information model and algorithms for trust decisions. It motivates the need for the different elements of a trust decision, then specifies how they are modelled in the trust management system. Related work presented in this chapter includes comparing TuBE with another risk-aware trust model, and evaluating how well the available policy languages for trust management can support the expression of the relevant policies in TuBE.

Chapter 4 focuses on reputation management, opening the third and final core area of research contained in this thesis. It defines the aggregation algorithms for receiving, interpreting, evaluating the credibility of, and consuming experience information through reputation systems. It also specifies the different points of policy configuration to direct the trust decision and reputation aggregation processes; these policies form the basis of making the system adjustable to different business situations. Related work on reputation management is presented in this chapter. First we study the common and differing elements in systems within the most significant class of reputation models, after which the interoperability between TuBE and different reputation systems is discussed.

Chapter 5 describes the TuBE trust management system architecture and prototype implementation. It first discusses the core system for trust decisions and incorporating new experiences, then extends the discussion to the overall architecture, with connections to external systems such as the Pilarcos monitor, reputation networks and configuration tools. The chapter finishes with an analysis of the configuration capabilities and input needs of TuBE.

Chapter 6 evaluates the TuBE trust management architecture through criteria presented in Chapter 2: conceptual usability of trust decision making, support for autonomy, adjustability for different business situations, implementation of social control, scalability and feasibility, and attack resistance. Related to this, it also presents the results of two simulations, with analysis of the behaviour of different policies for reputation-based trust decisions, and their robustness against attacks. Related work on simulation experiments for electronic commerce and reputation systems is presented in this chapter to demonstrate the state of the art in evaluating reputation-based systems, after which the overall feasibility of developing a general benchmark for trust management systems is discussed.

Chapter 7 concludes the thesis, summarizing the impact of the contributions made and discussing future work.

A glossary of Pilarcos and TuBE terminology, a trust management survey and a reputation management survey are included in the appendix.

Chapter 2

Trust management for inter-enterprise collaboration

Introducing reputation-based trust management into inter-enterprise collaborations allows service providers to flexibly discover new partners, and react to changes in the behaviour of previously known partners. The automation of routine decisions on joining and continuing in collaborations makes it feasible to ensure both that the estimated benefits of entering a collaboration outweigh the risks, and that this balance is kept throughout the collaboration itself. In addition, collecting reputation information on collaboration participants creates a social pressure to follow contracts, as good reputation will improve the chances of a service provider being chosen into collaborations.

This chapter sets trust management in its application area of inter-enterprise collaboration. It presents the research problem and outlines the requirements of the trust management solution. In the first section, we describe the operational environment of inter-enterprise collaboration, which forms the context for trust management processes. In the second section, we define the trust management architecture, specify how trust management processes connect to different phases of the collaboration, both in its establishment and operation. We also set the goals for the trust management system, including flexibility and attack resistance, which provide the basis for evaluating the system in later chapters. The third section discusses related work in inter-enterprise collaboration management, contrasting relevant existing proposals to our work.

2.1 Collaboration in open business networks

An *inter-enterprise collaboration* involves a group of enterprises in a contractually regulated, cooperative project. Collaboration allows the enterprises to focus their resources on a few key fields of expertise, while continuing to provide broader services for customers. It also enables small and medium enterprises to compete in fields dominated by large corporations by joining together to gain more influence than they would have separately. The enterprises maintain their independence during the collaboration, and make local decisions based on the enterprise policy.

Connecting different information and communication systems across organizational borders gives rise to interoperability issues, ranging from technical problems to business-level disagreements. The service-oriented architecture [Pap03, M^+06] aims to promote technical interoperability by exposing the different systems as service interfaces, which hide the internal implementation [BHM^+04]. Disagreements, on the other hand, must be handled through policy management and enforcement mechanisms.

Each enterprise provides a service to other members of the collaboration, according to a negotiated contract. The services involved are primarily regulated by local policy set by the enterprise, however. The regulation mechanism is independent from the core technical service application.

In a collaboration, services offered by different enterprises are connected to fulfil a more complex goal, such as a supply chain to refine raw materials into end products, or providing a value-added service. Each member of a collaboration has a particular role to fulfil, and does so through offering the service or services expected of the role. A logistics company could for example fulfil the role of a transporter by providing a service to order and pay for a transport, and possibly an additional interface to provide information on how much time the delivery is estimated to take.

The enterprises providing the services needed for a collaboration are located from the open service market; in other words, new service providers can enter the market simply by offering their service in it, and new types of services and collaborations can be added to expand and evolve the market. The market is also not centrally controlled by any single party, and service providers participating in a collaboration remain autonomous.

The services are loosely coupled: only the interfaces to access the services are exposed, while their internals are kept private. This division has multiple benefits. First, service providers do not need to repeat costly integration projects between each other to get new business networks into operation. It also makes it simpler to exchange one service for another if needed, without necessarily having to disband the entire business network.

Third, the service implementations can be based on different platforms, with different internal communication interfaces, as they only need to exchange messages between each other in an agreed-on way.

To realize this vision, we have developed the Pilarcos infrastructure services [KRRM08, KMR07, KRM07]. Pilarcos supports inter-enterprise collaboration management in multiple ways:

- finding the service providers matching service needs set by the collaboration,
- matching the proposed services together for interoperability,
- negotiating service provision terms acceptable to all parties, and
- monitoring both interoperability and contract compliance at operational time.

An overview of the division to common knowledge repositories, global infrastructure and local supporting agents is given in Figure 2.1. For a more detailed discussion, we refer to earlier work [KRRM08]; in the following, the central concepts of the operational environment are discussed to form the basis where trust management is incorporated to.

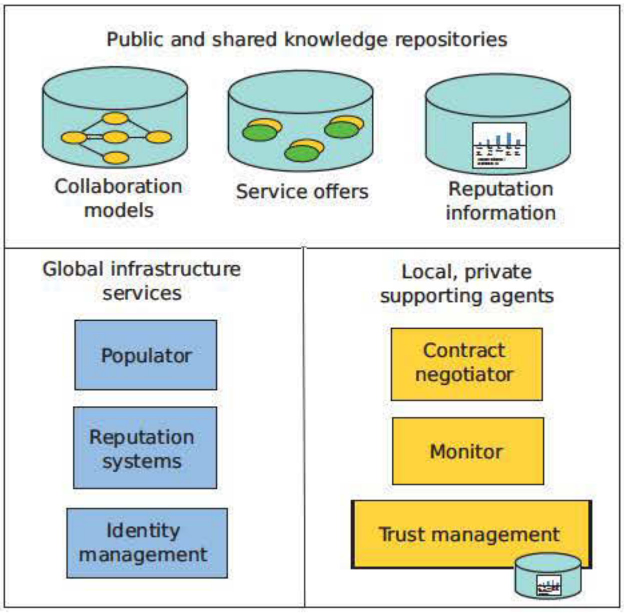

Figure 2.1: An overview of the Pilarcos architecture.

The concept of service is in the core of inter-enterprise collaboration. A *business service* represents the enterprise in a collaboration, and is managed by policy. It encompasses the actual *technical service application*.

The technical service application provides the basic functionality of the service. It can either be a computational interface to access a real-world service, such as a hotel reservation over the Internet, or it can be fully electronic, such as a money transfer or a customer information query. Both the computational and the real-world effects, such as transfer of physical goods, are a part of the service. The interface to access the service is well-defined, so that access can be automated in a collaboration.

The implementation of a service is typically general enough to be capable of a wide range of behaviours. In a collaboration, however, the behaviour of the service must be limited to the subset of activities that are reasonable in the collaboration context. For example, while it is a technically equivalent process to transfer ten or a million euros from one account to another, the latter event is quite unusual. While the former transfer can be routinely accepted, in the latter case the bank will want to carefully ensure the request is valid before allowing the transfer to happen. For this purpose, the technical service must be combined with monitoring facilities that allow or disallow certain kinds of behaviour to enforce internal policies.

A business service comprises the combination of the technical service and the monitoring facilities. Separating the technical service from policy enforcement makes it possible to easily adjust the policies according to the changing needs of both the collaboration and the enterprise providing the service. The service implementation itself does not need to be modified for this purpose; it is sufficient to reconfigure the policies enforced by the monitor. This structure of a business service is depicted in Figure 2.2.

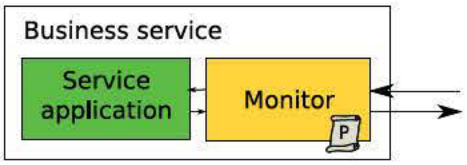

Figure 2.2: A business service consists of a technical service application and a monitor, which enforces local policy.

In Pilarcos, collaborations are represented as *business networks*, which comprise the roles of the collaboration and interactions between them, as well as the services fulfilling the roles. A *business network model* defines the roles through formal *service types*, and sets basic interaction rules between the services. Service types define the interfaces and requirements for the service as well as adjustable parameters for it. Information about the interoperation between specific service types is integral to support automated interoperability checking, done by the infrastructure services on establishing new collaborations [RK07]. The production of business network models and service types falls naturally to business consortia and standardization bodies: they implement laws and regulations governing the type of business in question, capture the best practices of a given business domain and guide its evolution.

Services implementing a particular service type are advertised in *service offers*. A service offer specifies the service type it fulfills and sets the various parameters for offering the service, such as quality of service guarantees, pricing, or the name of a specific transaction protocol to use. If some parameters are open to negotiation or depend on the proposed collaboration, the parameter values can be provided as ranges or a set of options. In this case, the final values become fixed when a contract between proposed service providers is being negotiated.

Service types, readily defined business network models and service offers are stored in corresponding *repositories* [KRRM08], as depicted in Figure 2.3. This metainformation allows new collaborations to be set up automatically.

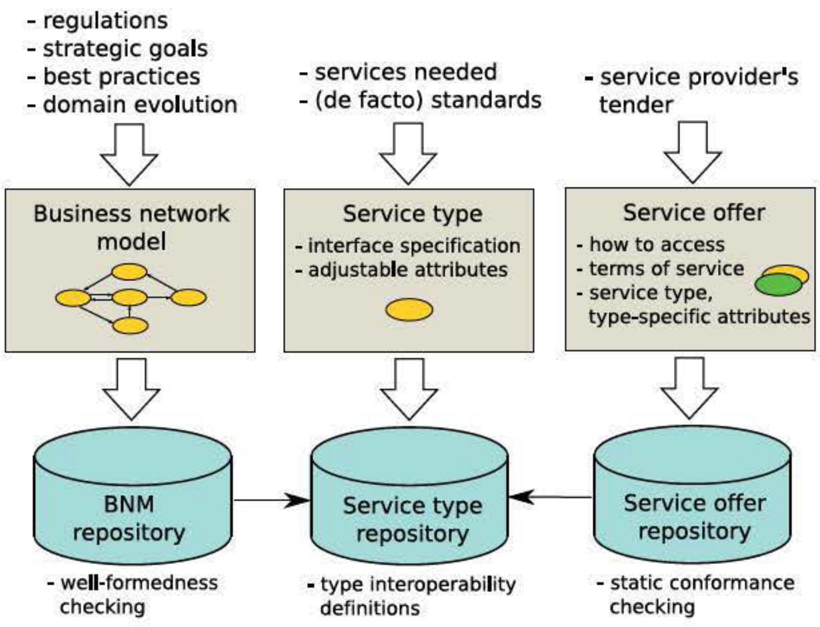

Figure 2.3: Business network models, service types and service offers support automated collaboration management.

Each repository also controls what kind of information is stored in it, for example checking the models for errors, ensuring that a service offer matches the format required by the service type it claims to fulfill, and checking that the identity of the service provider is traceable from the service offer. It is not the task of a repository to check if the service providers are trustworthy or that the service types are useful, however; these are choices made autonomously by the other service providers.

An inter-enterprise collaboration can be divided into five phases: *modelling*, *population*, *negotiation*, *operation* and *termination*. The different phases are presented in Figure 2.4.

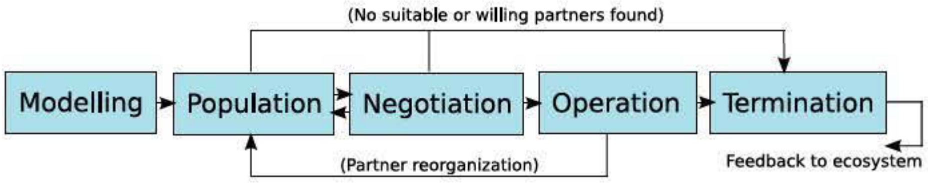

Figure 2.4: The phases of an inter-enterprise collaboration.

The *modelling phase* encompasses setting up the information needed to support the collaboration setup that we defined previously. The phase is completed separately from the other four phases, and the models produced are statically verified and stored into repositories. The resulting models can then be retrieved from the repositories to be reused in multiple collaborations. During the modelling phase, a domain expert applies service types to produce a business network model, describing the structure, roles and interactions of a business network. Using the same service types as a basis, service providers define and publish their service offers to advertise the services they provide.

In the *population phase*, an interoperable set of potential collaborators are discovered with the help of the information produced in the modelling phase [KMR07], as depicted in Figure 2.5. The Pilarcos collaboration management services include a *populator*, which automatically seeks service offers suitable for a business network model and checks them for interoperability. To begin a population phase, a business network *initiator* calls the populator service and passes it the business network model it wants to have populated with services.

The initiator also indicates how it wishes to participate in the business network itself by providing its own service offer, or offers, to the populator as a starting point; the populator assigns the offer to the desired role in the collaboration. The populator then proceeds to discover service offers

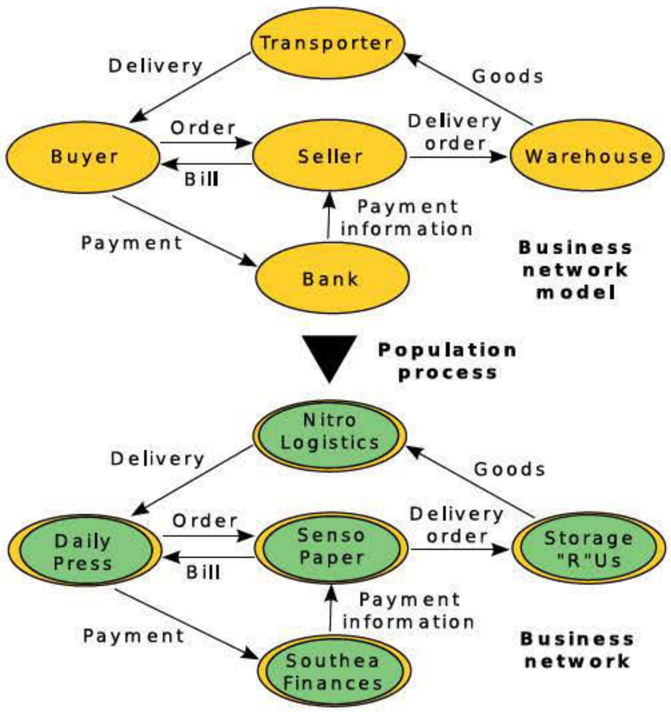

Figure 2.5: Populating an example business network from a model.

matching the remaining roles from the service offer repository, and performs static interoperability checking between the offers based on information about how different service types can interoperate.

In the end, the populator produces a set of readily populated business network proposals, out of which the initiator selects one for negotiation. In the example network, the Daily Press, shown as fulfilling the Buyer role in the network, would be a likely business network initiator: it has a need for the Seller's goods to meet its own business goals.

In the *negotiation phase*, the proposal is distributed among the other potential partners for approval, and any unbound terms, such as service pricing, are fixed to produce an *eContract*; an electronic contract representing the collaboration. The basic structure of the contract is determined by the business network model, and it contains references to both that and the adjusted service offers populating the business network. Rather than producing legally valid contracts from scratch, the automated negotiation only selects between and slightly adjusts pre-made contracts that have been prepared for the business network model.

The negotiation phase can involve multiple rounds of adjustments made to terms of service by different service providers. Not all information is published in service offers, as for example the pricing or specific protocols demanded by the service provider may depend on who it is proposing to collaborate with. Some service providers may refuse to participate in the business network outright. Even the initiator itself may disagree with some proposals produced in the population process; as the populator can be a third party service, it is not trusted with sensitive details such as which service providers are not acceptable as partners, or which ones are charged extra [KMR07]. If a proposed member refuses to participate in the collaboration or no mutually acceptable terms of service can be found, the initiator must select another business network proposal for negotiation.

In the *operation phase*, the collaboration has been fully set up and is ready to run. The business network model sets rules for what format the service invocations are to be made in, but it does not necessarily define a strict workflow as such. The operational phase of the collaboration may be a brief handful of transactions, or it can go on for months or years. During it, partners may leave or be removed from the collaboration due to failing to follow the negotiated eContract; if this is the case, the collaboration returns to the population and negotiation phases to try to discover and initiate a new partner to fill in for the missing one. Depending on the type of collaboration, this kind of reorganization is coordinated either by the business network initiator or another service chosen in the population phase to fulfill the role of coordinator.

Although the service offers have been statically checked for interoperability during the population phase, it is not a given that they actually conform to the defined behaviour during the operation of the business network. They may send service requests that do not follow the service type, fail to send required messages on time or send them in incorrect order. In a dynamic, complex environment of autonomous services, the possible causes for such contract and business network model violations are numerous: for example, the service may be temporarily overwhelmed by requests, the provider may be malicious or the service may have been hacked by an external attacker, the service software may be buggy, or the contract and the provider's local policy may be in conflict.

These kinds of contract and business network model violations are monitored for by each service provider independently, and a misbehaving participant can either be required to provide compensation for violations or be removed from the collaboration altogether.

Once the operational phase completes, either through successfully completing the tasks set for the collaboration, through having operated for a given time, or by running into an incorrigible state of error, it moves to the termination phase.

The *termination phase* ends the collaboration. When termination follows the operational phase, it involves controlledly bringing down the collaboration: for example, agreeing on post-collaboration responsibilities and releasing resources reserved for the network.

The collaboration may also be ended before it has begun: First, in the population phase, if suitable partners are not found, the collaboration ends prematurely. Similarly, in the negotiation phase, the business network proposals built by the populator may turn out to have members unwilling to collaborate with each other, or having too strict requirements on the eContract terms beyond the demands they were willing to express in the public service offer.

In all cases, the termination phase produces feedback to the service market, the infrastructure services and supporting metainformation: experience on how the collaboration went, the applicability and popularity of specific service types and collaboration models, or a need for new kinds of models.

2.2 Automating trust management between enterprises

All inter-enterprise collaboration builds on trust: the autonomy of the partners creates risks and uncertainty which can only be accepted when the participants have sufficient trust for each other. In this section, we define the central concepts and architecture for trust management, relate its processes to different phases of the collaboration presented in the previous section, and specify the goals of the architecture used as the basis of its evaluation in Chapter 6.

2.2.1 Computational trust in an open market

We define trust as *the extent to which an actor is willing to participate in a given action with a given partner, considering the risks and incentives involved.* The definition underlines the calculative nature of computational trust: some risks can be mitigated, while others cannot, and a *trust decision* determines whether the risks are tolerable for the given commitment. There are many alternative definitions for trust [RK05]; our definition is close to McKnight and Chervany's concept of trusting intentions [MC96].

A *trustor* is the actor making a trust decision, while the target actor of the decision is called a *trustee.* In both cases, the actors are business services. While trust as an emotional state or relationship takes place between human users, and to a degree groups of people in enterprises, we cannot attach computational trust directly to them. In inter-enterprise collaboration, the identities that we can reliably connect experiences to are bound to business services, and the service providers behind them can only be reached indirectly through the services. A given service may be offered by a coalition of multiple organizations, and a sufficiently large organization can provide hundreds of services across different domains; the services may behave inconsistently simply due to not being centrally controlled within the enterprise.

Trust management is the activity of upkeeping and processing information which trust decisions are based on. Specifically, we do not include the more general concept of managing business relationships in our definition. Trust management interacts closely with risk management in an enterprise. Whereas risk management aims to identify and mitigate risks, trust management brings together the estimated risks with organizational policy in a given decision context. A trust decision considers the incentives that push for a positive decision despite the risks. The risks estimated in a trust decision are ones that either cannot be mitigated or are not always worth the cost of mitigating: in other words, they are risks that must be tolerated in order to collaborate.

We can identify three prevalent approaches to inter-enterprise trust management in related work: strategic networks, certification, and experience-based approaches. Approaches based on strategic networks are direct descendants of the traditional method of building trust over the years between human representatives of enterprises. They rely on forming a single domain of trust, a closed *virtual breeding environment*, from previously known partners. Before a service provider is allowed into this environment, it must be manually evaluated for trustworthiness, but after being allowed into the virtual breeding environment it is fully trusted and can enter into collaborations with other members. TrustCoM [$\mathrm{W}^{+}06$], ECOLEAD [RGAN06] and CrossWork [MG10] are examples of this type of approach; we return to the first two systems in Section 2.3, while CrossWork represents the extreme case of no explicit trust management: the set of strategic partners is fixed and fully trusted.

Certification-based approaches also depend on manual trustworthiness evaluation, but rather than depending on a single trust domain, each service provider evaluates for itself whether it accepts a given set of certified credentials. The credentials can indicate for example membership in a consortium or a local group of businesses, and they are exchanged in two-way negotiations between the service providers. The certification-based approach has emerged as an extension to traditional access control, and is mostly represented by policy languages; WS-TRUST [OAS07], PolicyMaker [BFL96] and TrustBuilder [WSJ00] represent this approach. Section 3.3.2 discusses policy languages in more length.

In experience-based approaches, trustors evaluate the trustworthiness of other providers based on previous experiences with them, and update their view based on new experiences. The generation of experiences can be automated and incorporated into the trust management system, which is a distinctly new trend in inter-enterprise collaboration. The roots of experience-based approaches are in electronic markets; CONOISE [JI02], SECURE [C$^+$03] and Regret [SS02] are examples of the experience-based approach. We will present CONOISE in more length in Section 2.3.2, SECURE in Section 3.3.1, and return to other related approaches in Section 4.3.

Trust management in strategic networks is centralized and static: an ultimately trusted party must keep track of the members of the breeding environment, and any changes in memberships require manual intervention. Certification-based trust management is also static, as issuing and revoking certificates is done outside the system, but management can be distributed through delegation. Experience-based trust management is dynamic and distributed, and it provides support for trust decisions throughout the different phases of the collaboration.

Before we discuss the factors influencing a trust decision further, we must define the setting in which trust decisions are made. The context in which computational trust is used determines what kind of trust model is needed.

We make a set of background assumptions on the nature of typical collaborations supported by interoperability middleware. Given the competition in an open market, and the relative ease of setting up flexible collaborations, we consider the typical business relationships behind the business networks to be relatively short-term, rather than relying on years of natural trust building. The business value involved in the collaboration varies, but single collaborations are typically not sufficiently important to prove critical to an enterprise. All this means that collaboration decisions can afford to be somewhat opportunistic: optimistic collaboration with partners that have not been known for a long time brings new business, but it also comes with new risks.

An opportunistic market solicits enterprises to stretch contracts and look for loopholes. For example, while a contract is likely to have a legally binding compensation clause to discourage not delivering bought goods, it may not have much of a punishment defined for a late delivery or cutting corners with the quality of the goods.

The central approach to operating in a world like this involves entering into collaborations with some care, evaluating the risks involved, and learning from mistakes. Enterprises will have some incentive to share their experiences, because this helps keep malicious partners from altogether ruining the market [RZFK00, JHF03], but they may also omit information or spread false information if it suits their goals. Due to its flexibility, the experience-based approach to trust management is the best solution for this type of environment.

There is no centralized control or monitoring that all collaborators can rely on, hence all enterprises must be prepared to make local decisions on whether to collaborate, based on information they have gathered themselves and from the network. The enterprises are also free to decide that they will not follow a contract when it is not in their best interests; compensation clauses are defined to set the price for violating an agreement.

2.2.2 Automating trust decisions

Trust decisions evaluate and weigh the estimated risks and benefits of collaborating. The estimations are based on experiences: collaborators who do not respect agreements are likely to not respect them in the future either, and should therefore not be chosen again if better options are available. As trust decisions are made locally, based on local information and enterprise policy, different actors may arrive at different conclusions on the same trustee. In addition, the action which is being decided on has a strong effect on the outcome of the decision; some actions require more resources or involve more inherent risk than others.

Our goal is to automate *routine decisions* where the balance of risks and benefits is clear. We do not aim to automate all trust decisions, but instead leave borderline cases and decisions involving very high stakes to human management.

The main problem in automating trust decisions for inter-enterprise collaboration is that the information to support them is currently not available in a form that can be automatically processed. Conceptual work has been made [RK05, JIB07] and some general models have been proposed, but there are no trust decision systems in actual use between enterprises. We will discuss existing trust-aware inter-enterprise collaboration management research in Section 2.3, and return to trust models in more detail in Section 3.3.

A good trust decision depends on a good risk estimation. Risk estimations are predictions, setting probabilities for different kinds of outcomes from a collaboration. These probabilities are estimated based on information about past events, experience from earlier collaborations. This information can be first-hand experiences, shared experiences, or a combination of the two. A central source of such experience is a reputation system.

We define reputation as *the perception an actor creates through past actions about its intentions and norms* [MMH02]. A *reputation system* collects, processes and distributes reputation information, such as concrete experiences or more general statements about an actor or its trustworthiness [RKK07]. It is implemented by a *reputation network*, which encompasses a reputation system and a network of actors sharing their experiences within it.

A reputation system brings a new role into a trust decision besides the trustor and the trustee: that of a *recommender*, a source of external reputation information. The external reputation information is passed as *recommendations*, which, despite the name, can be either positive or negative. Experience-based recommendations can be direct, coming from the actor who has entered into a collaboration with the trustee in the past and transacted with it, or they can be indirect, in which case the recommendation passes through middle-men before reaching the trustor. The roles of actors in a reputation system are depicted in Figure 2.6.

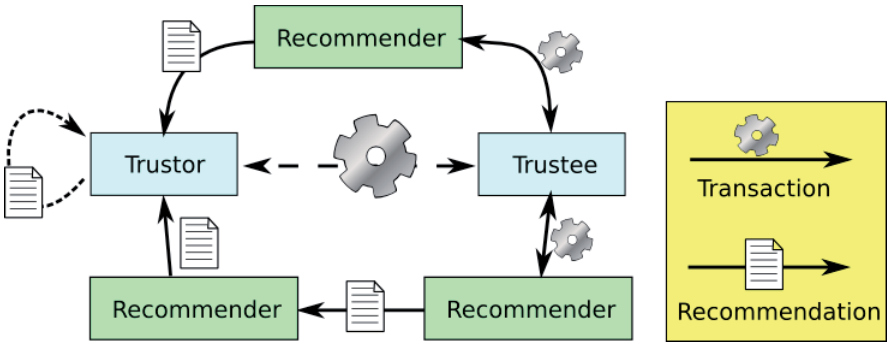

Figure 2.6: To decide whether to perform a transaction with the trustee, the trustor collects experiences on earlier transactions, including its own.

Trust decisions are a part of the policy setup enforced by the monitors we presented in Section 2.1. The monitors apply both local and shared policies; the latter are represented in the eContract. If the local policy and eContract disagree, local policy is followed.

Similarly, trust decisions are based on a combination of local and shared experience information, where local experiences take precedence. Besides experiences, trust decisions take other local information as input as well, but on a high level these can be seen as a part of the policy.

Trust decisions in inter-enterprise collaboration have certain characteristics that set them apart from, for example, trust decisions between private people. The main difference is the infrastructure investment that can be assumed from organizations.

The partners in a collaboration enter into legally binding contracts, which means that if compensation clauses are not followed, the underlying legal system can be expected to step in to enforce the rules in the end, even though the process may be time-consuming and not bring back the lost resources. In order to negotiate and sign such contracts, the partners must be strongly authenticated. We assume sufficient infrastructure to exist for this, and will not elaborate on it further.

The electronic contracts also provide central support for ensuring that partners contacting a business service are authorized to use the service. The decision to make, then, is solely on whether there is sufficient incentive to enter and continue with the collaboration or not. We have discussed the broader policy environment and automation of decision-making on different levels more extensively in earlier work [KRM08].

2.2.3 Trust management in collaboration phases

Trust management takes on varying roles in different phases of the collaboration. These differences can be divided into two categories based on the two central tasks of a trust management system, information processing and making trust decisions. Figure 2.7 contrasts the trust management process with the collaboration phases presented earlier, describing the information made available in a given phase, and the types of decisions needed in it.

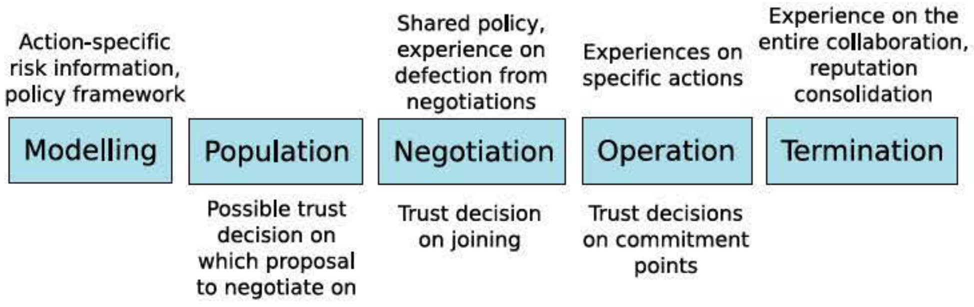
Action-specific risk information, policy framework
Shared policy, experience on defection from negotiations
Experiences on specific actions
Experience on the entire collaboration, reputation consolidation
Modelling
Population
Negotiation
Operation
Termination
Possible trust decision on which proposal to negotiate on
Trust decision on joining
Trust decisions on commitment points

Figure 2.7: Trust information and decisions in different phases of the collaboration.

In the modelling phase, domain experts produce business network models and enterprises define service offers. The business network models in particular are a natural target for annotations: they can be augmented with information to support decision-making, such as explicitly expressing the cost of not following a contract at a given point, or providing a formula for the scale of risk inherent in a specific transaction.

The business network model provides a framework for the shared policy that is eventually refined in the negotiation phase and stored in an eContract; the domain expert is also in a good position to estimate, for example, how the financial risk involved in buying goods depends on the price of the item bought. While this information might not be trusted or found useful by all users of the business network model, having it readily available in a repository allows it to be used to enhance trust decisions without having to perform the modelling separately within each enterprise.

No explicit trust decisions are made in the modelling phase; the enterprise does need to determine how specific it wishes to be in its service offers, but we find that these kinds of evaluations are not routine or straightforward enough to lend themselves well to automation.

In the population phase, the business network initiator may wish to make a trust decision on which of the produced proposals it wishes to present for negotiation. In terms of trust as a willingness, the decision measures which proposal has the best balance of risks and benefits, based on the proposed participants [KMR07]. The population process combines a set of service models into a joint policy framework, but it does not produce new information for trust decisions as such.

In the negotiation phase, the policy framework set in the business network model is refined, and for example the compensation required of partners who choose to not follow the contract becomes fixed. The threat of compensation provides one of the incentives to participate in an action, and avoiding compensation should be taken into account as one of the certain benefits of a positive trust decision.

During the negotiation phase, all potential participants will make a trust decision on whether they are willing to join the proposed collaboration. The decision may also affect the special conditions which the service provider may set for joining, and it may be repeated at the end of the negotiations.

The negotiation phase is a first point of where the members of the collaboration can observe each others' behaviour. A service provider may publish a service offer solely for the purpose of being invited into negotiations with its competitors and to study what their actual, non-public terms of service are. If it has no intention to follow through with the collaboration, it can misbehave in the negotiations or withdraw at the last minute. This kind of behaviour cannot be made impossible without reducing the autonomy of the participants, but it can be made less attractive through sharing experiences and warning other actors of questionable behaviour.

The operation phase is the main target of monitoring and trust management; even though the eContract gives some idea of who the collaborators are and what kind of activities can take place, it does not determine the actual behaviour of the actors once the collaboration begins. The sources of input to the monitoring process are depicted in Figure 2.8.

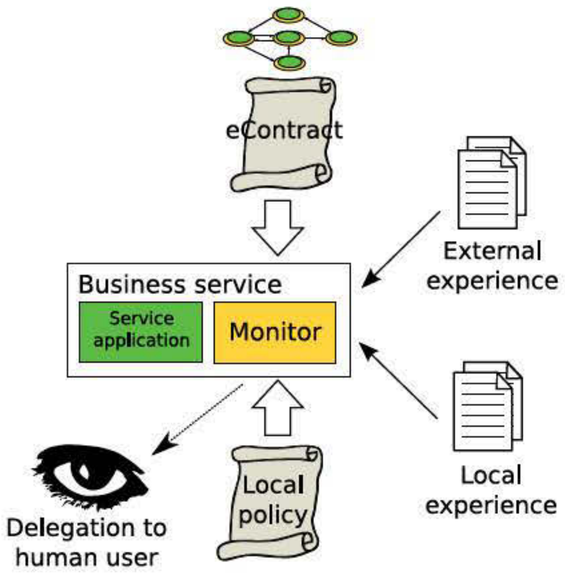

Figure 2.8: Monitors enforce local and shared policies, including making routine trust decisions based on local and shared information. Non-routine cases are delegated to human users.

Trust decisions are made whenever resources, such as funds, processing capacity or production materials, are being committed. After each relevant transaction, the actors store local experiences on whether the transaction completed satisfactorily. These experiences in turn influence later trust decisions, both within the same collaboration and in future collaborations. In some cases, transactions cannot be readily considered as completed before the collaboration is finished; in this case, the experiences are uncertain up until the termination phase.

In the termination phase, the outcome of all transactions is known, and the overall outcome of the collaboration can be evaluated. The kind of consolidated, high-level experience information that can be produced at the end of a collaboration may also be more valuable for sharing in a reputation system than very detailed transaction experiences; it is also less likely to inadvertently leak information about the internal processes of the service provider.

2.2.4 Architecture goals

In the earlier sections, we have defined a trust management extension to the service ecosystem architecture. We expect the enhanced architecture to have the following properties:

- an information model for multi-dimensional trust decisions,
- the algorithms and specific points for making the trust decisions privately within each enterprise,
- the aggregation algorithms for receiving, interpreting, evaluating the credibility of and consuming experience information through reputation systems to ensure that the architecture learns from experience, and
- support for adjusting to different business situations within the enterprise through private policies and metapolicies.

The information model and decision-making algorithms are presented in Chapter 3, while reputation aggregation and policy configuration points are discussed in Chapter 4. We demonstrate the implementability of the architecture with a research prototype presented in Chapter 5.

In Chapter 6, we evaluate the architecture based on six criteria, which arise from the needs of the application area of inter-enterprise collaboration:

- the conceptual usability of trust aspects in decision making,
- support for autonomy,
- adjustability for different business situations,
- implementation of social control,
- scalability and feasibility, and
- attack resistance.

The criterion for providing conceptually usable decision making tools underlines the necessity for using concepts that are consistent with the domain where policies are set: for defining decision policies on business services, business-level concepts are considerably better suited than implementation-specific technical concepts. The trust management system should map business-level concepts to implementable technical concepts in a way that provides sufficient expressive power to define effective trust management policies.

Supporting autonomy encompasses the need to allow services to make their own decisions, to be allowed to control their own information and to minimize the need for submitting them all to centralized monitoring or rule enforcement.

Adjustability for different business situations connects to the architectural requirement for providing ways to adjust the trust decisions. We analyze whether these ways provide sufficient support for the system to evolve gracefully as the needs of the organization change.

Implementing social control in the open service ecosystem entails that in the absence of a central authority to enforce rules, contract violations and other misbehaviour can be sanctioned in a distributed way. In order for this to work, the sanctioning system must operate on two levels: punishing regular, i.e. first-order misbehaviour as well as sanctioning any unfair punishments (including lack of deserved punishment) to first-order misbehaviour, on the second level. This combination allows the community represented by the open service ecosystem to remain operational even as it grows in size.

The criterion for scalability and feasibility of a system encompasses the requirement that the costs and benefits of implementing such a system must be in balance. Costs can be divided into two categories: computational costs encompass how heavy the system is to operate in terms of algorithmic complexity, messaging and storage, while administrative costs comprise the necessary human labour to set up and operate the system.

The attack resistance criterion reflects a balancing between two goals. On one hand, the trust management system should be resistant to and able to punish different forms of misbehaviour, including coordinated attacks. On the other hand, the system should remain usable and not be so strict that it would start to severely inhibit business in comparison to human decision-making.

When selecting our criteria, we found many matching criteria in related work, as well as some that were even opposed. For example Dingledine et al. have specified some desirable properties for a reputation system [DFM00];

their focus is on reputation systems only, and as a result the criteria are somewhat low-level, such as distinguishing between a complete lack of experiences and negative experiences recording bad behaviour. They present several criteria related to usability, scalability and feasibility and attack resistance. The list also includes a criterion in direct conflict with our goals, stating that no one should be able to learn how a given actor rated another actor except the rater himself. While the requirement aims to protect actors from retribution for reporting misbehaviour, it also removes all basis for credibility analysis or ensuring honesty in ratings. It becomes apparent that when comparing criteria for different reputation systems, it is important to distinguish between the goal of providing a channel to anonymously alert investigative authorities of e.g. rule violations, subjective recommender systems for endorsing past business partners, and systems aiming to create social control through distributed monitoring and objective criteria.

2.3 Related work on trust-aware inter-enterprise collaboration management

In this section, we contrast comparable systems for inter-enterprise collaboration management to our work. As the thesis focuses on the trust management facilities, we will focus on work similar to the combination of Pilarcos and TuBE here; examples of comparisons to Pilarcos without TuBE can be found in our earlier work [KRRM08]. We present two examples of trust-aware collaboration management work that are similar to our efforts, and form a pair of representative samples of the state of the art: TrustCoM and CONOISE. The different approaches represented by these two examples are then contrasted to our proposal. TrustCoM is an Integrated Project in the European Commission Sixth Framework Programme, while CONOISE is a British collaboration.

ECOLEAD, also an Integrated Project in the Sixth Framework Programme, discusses aspects of inter-organizational trust at length [MAH+06, MA07], while its aim is in providing a set of tools for virtual breeding environments rather than an automated management architecture. We will therefore not discuss ECOLEAD further here.

2.3.1 TrustCoM

The TrustCoM project aims to build a framework for trust, security and contract management for collaborations that are implemented as dynamic virtual organizations. The TrustCoM framework includes a virtual breeding environment, service discovery and negotiation support, and trust and reputation management [W$^+$06].

The TrustCoM trust management service is primarily certification-based, as opposed to the experience-based approach chosen in this thesis. In TrustCoM, security tokens are issued to actors entitled to access the service. The tokens act, in essence, as certificates of trustworthiness. More specifically, the tokens represent attributes that indicate trustworthiness, such as membership in a trusted group of a partner organization [LS$^+$05]. This kind of separation of concerns between certification issuing and interpretation builds on attribute certification; see for example SPKI/SDSI [Ell04, Ell99] and comparison to identity certificates [Kar03].

Each actor may hold numerous security tokens. To support the actors' ability to control access to information about themselves, TrustCoM contains a trust negotiation component, which in essence helps the actor determine the tokens necessary for accessing the service. This makes it possible to only disclose the needed tokens, rather than all of them. Privacy-preserving trust negotiation strategies are an important research field in their own right; the problem has been explored by e.g. Winsborough et al. and Winslett [WSJ00, Win03].

Contrasting to Pilarcos and TuBE, we expect that security tokens will be a useful method for determining which actors are allowed to make service requests. Our experience-based approach assumes that a basic authorization service is in place, and focuses on protecting the system from authorized actors as well as determining who becomes authorized through a collaboration contract in the first place.

The TrustCoM reputation service is used in the context of finding suitable partners for a collaboration [LS$^+$05]. For example, potential partners with a reputation below a specified value will not be applicable. Experiences are gathered at the end of each step in a business process, and stored centrally by a trusted third party. A downside of relying on a trusted third party is that all participants must be able to trust it to remain impartial; on the other hand, if such a party can be found, it makes access to the reputation information very simple.

In TrustCoM, experiences do not affect ongoing collaborations. In comparison to our trust management system, the cycle of reaction to new experiences is prolonged from relevant transactions to entire collaborations, which may consist of hundreds of transactions. This tradeoff between low decision-making overhead and fast reaction is less drastic if we assume that all collaborations are very short, although this also indicates greater overall overhead in setting up the collaborations. In TuBE, we allow decision-making overhead to be lowered flexibly by making the set of relevant transactions more sparse, but using transactions as our basic unit rather than entire collaborations makes it more easy to add new checkpoints without adding to the overhead of setting up a collaboration.

In summary, the services provided by TrustCoM have goals that overlap partially with the Pilarcos collaboration management middleware in general, and to a degree with the TuBE trust management system specifically. TrustCoM does take advantage of reputation in partner selection similarly to TuBE, but its overall trust management approach is certification-based, while TuBE is experience-based. Defining a single set of trusted actors, like TrustCoM does, imposes a scalability limit to the ecosystem: new members must be found trustworthy before they can be introduced into the ecosystem. Certificates are used to express membership in the service ecosystem, while their issuance is outsourced. In contrast, Pilarcos and TuBE allow unknown actors to enter into the ecosystem and their trustworthiness to be evaluated through collaborations, as the system itself produces the experience information that determines which actors are likely good choices for collaboration partners, and which actors should be avoided.

2.3.2 CONOISE

The CONOISE project aims to build infrastructure to support robust and resilient collaboration formation and operation in dynamic, open and competitive environments [NPC$^+$04, PTJ$^+$06]. The grid-enabled CONOISE architecture includes service discovery and bidding support, quality of service monitoring and assessment, a policing service for solving disputes and references to experience-based trust services as future work. Jennings et al. have later done considerable work on reputation management in multi-agent systems, although it is not explicitly tied to the CONOISE architecture as such [TPJL06, HJS06b, TCRJ08]. We will return to related work on reputation models in Section 4.3.

In CONOISE, population and negotiation are done in a single process: first, the initiator of the collaboration asks potential service providers for bids for service provision; second, it assesses the quality and trustworthiness of the received bids; and third, it receives an optimal combination of service providers from the cleared set, generated by a population service. The initiator is the central choreographer of the collaboration: it interacts with the customer directly, and is responsible for ensuring that each member in the collaboration provides the agreed service.

The main downside to a single-round population and negotiation is that service providers must set their bids for participating in a collaboration without knowing the other participants. While this is not necessarily a problem for centrally choreographed systems with a closed set of available bidders, it causes problems with the Pilarcos assumption of an open market: as service providers may have strong preferences on which other providers to work with, there is a need for a separate negotiation round.

During the operation phase, a quality of service monitor in CONOISE observes the traffic between service providers and the customer and reports to the initiator. If the monitor detects a breach of contract or that the quality of service drops below acceptable, it alerts the initiator, who then begins a repopulation process to replace the service provider at fault. The initiator stores this as experience on the service provider [PTJ$^+$06]. For sharing experiences, information about the observed quality of service is combined with information about the expectations of the service providers sharing their experience [NPC$^+$04].

The CONOISE monitoring model is centralized: a single agent is able to observe the entire collaboration. It fits with the model of an initiator-centric collaboration. The more pointedly peer-to-peer collaboration model of Pilarcos, due to maintaining the autonomy of the participants, makes global monitoring unfeasible. Members of the collaboration can and do share some state information in order to keep the collaboration running, but by default, experiences are formed from local observations, and external information must be separately evaluated for credibility. Quality of service is a central target of monitoring for gathering experience information in Pilarcos and TuBE as well.

The CONOISE view of sharing information both on observations and expectations is compatible with the TuBE trust management system, which models the fulfilment of expectations as a specific type of observation. We will discuss fulfilment and objective effects further in the next section.

In terms of openness, CONOISE seems to follow the traditional grid philosophy of a quite fixed set of service providers and available services. Within this set, it supports evidence-based selection between the available service providers, and reacts to misbehaviour during the collaboration.

2.4 Chapter summary

In this chapter, we have described the open service ecosystem, a service market environment supported by a global infrastructure. The open service ecosystem consists of repositories of public and private knowledge encoded in metainformation, such as the service types, business network models and other common vocabulary, service offers from service providers, and experience information to support trust decisions. The global infrastructure supporting the ecosystem is built on top of this metainformation, including service discovery and selection, eContract establishment, monitoring and trust management. Within this ecosystem, a number of collaborations simultaneously exist in different operational phases. The building blocks of the open service ecosystem are summarized in Figure 2.9.

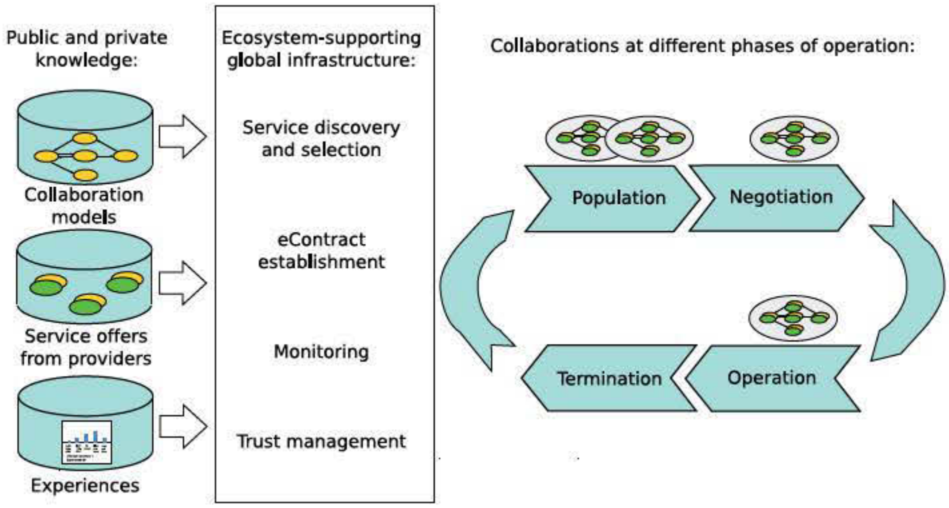
Public and private knowledge:
Collaboration models
Service offers from providers
Experiences
Ecosystem-supporting global infrastructure:
Service discovery and selection
eContract establishment
Monitoring
Trust management
Collaborations at different phases of operation:
Population
Negotiation
Operation
Termination

Figure 2.9: The open service ecosystem.

We have proposed to extend this construct with a trust management system, which automates routine decisions on whether the benefits of a collaboration outweigh the risks, and the processing of information needed to support these decisions. Any non-routine cases are left for the human user to decide on. The system should learn from first-hand and shared experiences, and be adjustable to different and changing business situations. We have positioned the trust management processes within the different phases of the collaboration and the division of responsibilities within the service ecosystem.

We have also set goals for the trust management system architecture to provide conceptual usability for trust decision making, support the autonomy of services in the ecosystem, provide adjustability for different business situations, implement social control and to be robust against attacks. These goals will set a guideline for the following chapters, which present the information model for trust decisions, the algorithms for private trust decisions and the aggregation of experience information, and the support for adjusting the behaviour of the system through private policies and metapolicies.

Chapter 3

The dimensions of a trust decision

In order to provide flexible automation for trust management, we must define an explicit mechanism for making trust decisions and for collecting the experience information to support them. This chapter presents the TuBE trust information model and algorithms for trust decisions. The first section motivates the different elements of a trust decision, while the second section presents the information model and how a decision is processed from the data. The third section presents related work on modelling trust and trust decisions, including a comparison with a risk-aware trust model designed for pervasive environments, and studying how well the state of the art in policy languages for trust management supports the needs TuBE has for policy expression. To complete the trust management process with automated learning from experience, Chapter 4 focuses on the feedback loop provided by reputation information. It also discusses policy configurations in more detail.

3.1 The trust information model

The goal of automating routine trust decisions requires a representation of the information that affects the decision in a way that can be automatically processed. In Section 2.2.4, we identified risks to assets and incentives, as well as first-hand and globally shared experiences as information to take into account when making the trust decision. In addition, the situation the decision is made in must be somehow represented. These aspects must be present in the trust information model. This provides a good match with human and organizational trust decision-making [Kau11, MC96, MAH$^+$06].

Trust decisions are made to protect assets. The TuBE trust information model encompasses a definition for the assets protected by trust decisions, and the five decision factors of *risk*, *risk tolerance*, *reputation*, *importance* and *context*.

Trust decisions are based on a calculative comparison of the estimated risk and determined risk tolerance. The risk estimate is built on a view of the reputation of the trustee, and tolerance builds on the importance of allowing the guarded action. Context adjusts all four other factors according to temporary changes in the situation the decision is made in. The interdependencies of the factors are summarized in Figure 3.1.

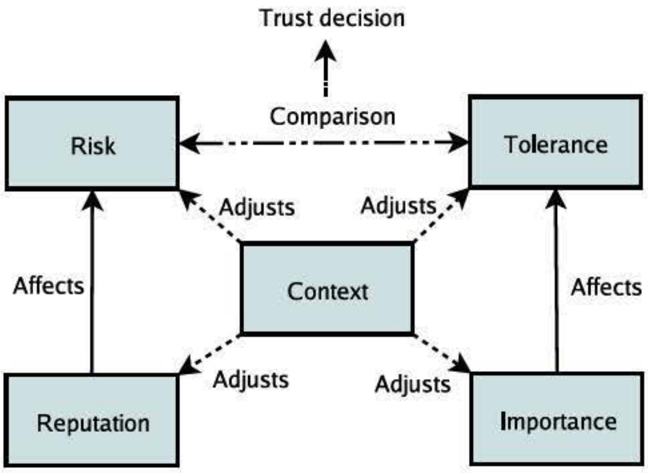

Figure 3.1: Factors used to form a trust decision.

3.1.1 Assets guarded by trust decisions

The TuBE trust management model expresses potential outcomes of positive trust decisions in terms of their effects on different assets the trust management system aims to protect. A simple approach would be to merge all assets into one, e.g. money, but introducing multiple high-level assets has the benefit of added clarity: it is difficult to convert effects such as reputation loss or gain, human injury or a security breach to monetary terms.

While the assets identified for protection vary, we believe that all enterprises will have use of a handful of high-level assets. Defining a standard set also has the benefit of increased interoperability: Experiences based on these assets can be used across systems with less information lost due to unmatched or vaguely defined assets. Privately defined assets may have a clear and valuable role in one enterprise, but not be understood the same way in another.

We have settled on a set of four standard assets: *monetary*, *reputation*, *control* and *fulfilment*. The monetary asset represents money and other artifacts in the enterprise that have a well-defined monetary value. The reputation asset represents the trustor's good reputation. The control asset is a joint representation for the trustor's security, privacy and general self-protection assets. The fulfilment asset represents the fulfilment of the trustor's expectations of the trustee's participation in the action, such as the quality of the service the trustee provides or its efficiency in fulfilling its end of the agreement. The assets and their associations are depicted in Figure 3.2.

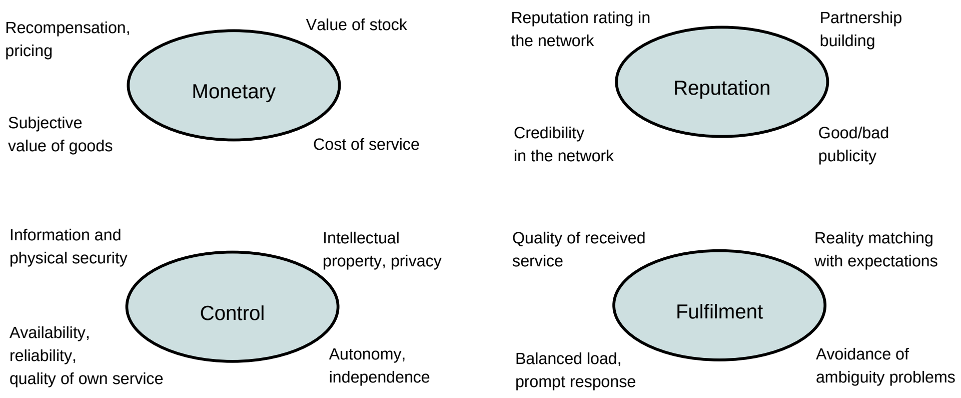
Recompensation, pricing
Value of stock
Monetary
Subjective value of goods
Cost of service
Reputation rating in the network
Partnership building
Reputation
Credibility in the network
Good/bad publicity
Information and physical security
Intellectual property, privacy
Control
Availability, reliability, quality of own service
Autonomy, independence
Quality of received service
Reality matching with expectations
Fulfilment
Balanced load, prompt response
Avoidance of ambiguity problems

Figure 3.2: The four standard assets.

The *monetary asset* forms the basis of decision-making in many situations, and it is also the simplest to measure. Even so, the monetary value of an item or a concrete service is not always straightforward to determine. The trustor can form a subjective value for a target by deciding how much it would be ready to pay for it. For trust decisions, this measure is more valuable than the actual current market value of such a target, and simpler to determine. If, for example, the trustor is unaware or uninterested in a particular feature of a device it wishes to buy, it is irrelevant to the decision whether the device carries such a feature or not or whether the feature works, even though the selling price of the device might depend greatly on it.

The *reputation asset* encompasses both the enterprise's reputation rating in any particular reputation system, as well as the more abstract notion of its public relations, positive appearance in the media, and the attitudes of its partners and customers towards it. When approached this broadly, a change for the reputation asset can never be measured as accurately as a monetary loss or gain, but it allows the enterprise to represent the risk of losing partners through a traditional drop in perceptions, even if for some reason the actual calculated reputation does not change. For a well-informed decision, the enterprise should generate both computational and more general estimations of lost reputation for particular action outcomes instead of simply monitoring a reputation network for changes which may not be directly connected to particular actions to begin with. As an example, a partner spreading misinformation about the trustor service threatens this asset.

The *control asset* represents the general need for an enterprise to protect itself from outside influences: to upkeep control over its security, privacy and other aspects of its autonomy. The security of an enterprise involves the physical safety of its people, equipment and goods, and less tangible aspects such as the continuity, availability and reliability of its services and the protection of its information and IT systems. The privacy of an enterprise encompasses its ability to control information concerning itself. It encompasses more than simple confidentiality: even if information becomes unconfidential when it is given to a partner, the partner can still violate the enterprise's privacy by passing the information on without permission or by releasing false information of its own. A reputation system is a privacy tradeoff in itself, and it can cause further privacy threats if a trustee is wont to release unfair or false experience information about its partners after collaborations. This asset is also threatened when a partner provides false reputation information *to* the trustor service through a reputation system, in order to skew its view of other service providers; in contrast, spreading misinformation *about* the trustor service threatens its reputation directly. Finally, the enterprise may feel its autonomy is threatened by some forms of collaboration, for example if an offered contract has severe enough compensation clauses to force local decisions to follow the contract.

The *fulfilment asset* is tightly connected with a trustee. It describes whether the trustee does its part of what was agreed, leaves something relevant undone or does something it was not strictly expected to, in the positive sense. Where the base item for the monetary asset is the wealth of the organization, the base item for fulfilment is the general trend of respected agreements, which is reflected on the success of the organization. The asset has high value in evaluating the predictability of the trustee over a range of highly different actions. Like more traditional assets, it can be protected: leaving things undone can be avoided by putting effort in negotiating a more tightly binding contract, or by selecting trustees more carefully, with weight given on their earlier performance. Reliability, quality of service or competence cannot fully be captured by the other assets: for example, if large deals with a particular trustee always end in less profit than was expected, but do not result in losses per se, the trustee has a spotless reputation money-wise, even if it is not quite as attractive a partner as a more reliable enterprise.

The fulfilment asset is a more explicit formulation of the broadly-accepted idea that one of the main aspects of trustworthiness is doing what was agreed upon. The eBay reputation system [eBa11], as well as any reputation system based on a simple model of cooperation or defection, measures only the trustor's subjective fulfilment of the deal. The fact that this outcome measure was the first one to be adopted to reputation systems speaks for its central nature. The first three assets, on the other hand, represent the concrete effects to the business, which makes them central to any enterprise's risk management, and an important factor in a trust decision as a result. The fulfilment asset can help with controlling the load placed on service providers with good reputation [RKK07]: once they either turn down proposals or respond to them sluggishly due to high load, they will cause worse experiences related to the fulfilment asset, and a trust decision system can be adjusted to react to this kind of development quickly.

As assets are used to represent outcomes of actions, each asset is connected with a five-step scale of the effect the outcome has on it. The middle step represents no change, while the two steps on both sides represent minor or considerable positive or negative effect, respectively. For the monetary asset, it is clear that a small company will focus on smaller amounts of money than a large one, and therefore the exact numbers to divide all real values to these five groups will be decided within each organization. This subjectivity is repeated over the other assets as well, albeit not quite as pointedly; differences between actors, their values and expectations complicate the semantics of all experience sharing, which must be taken into consideration in reputation systems design and when using information provided by them. For the control asset, it may be that none of the positive outcomes are ever used when storing experience, but they are retained for symmetry.

The described set of standard assets can be extended either by defining further, separate assets, or by refining the set of four. Sub-assets would have clear enough semantics to be shared across organizations, while new, separate assets are either more problematic or impossible to match.

3.1.2 Risk

The risk involved in a positive trust decision depends on the nature of the action and a prediction of how the trustee is likely to respond to a positive decision. A risk evaluation is an attempt to partially predict the costs and benefits resulting from different possible outcomes, which are expressed with the help of the asset model described earlier. The evaluation assigns these effects simple probabilities based on the observed costs and benefits of earlier transactions. An example risk evaluation is depicted in Figure 3.3.

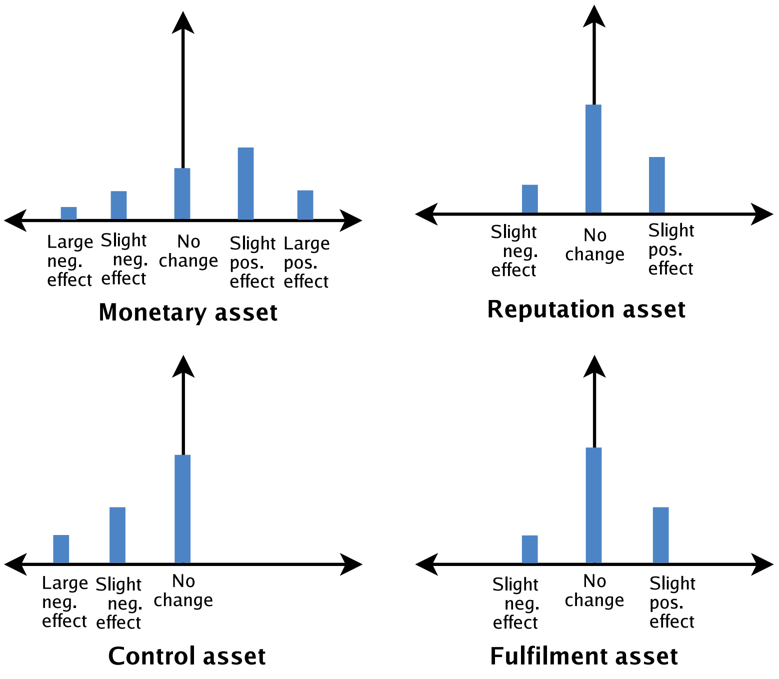

Figure 3.3: An example risk evaluation.

Risk depends on an estimation of how the trustee behaves, and the openings for costs or benefits created by the action. The former is built from the reputation of the trustee in the trustor's system, and is updated as more experience is gained on the trustee. The latter depends on the type of action and some of its parameters, which define the range of possible outcomes for the action. For example, when arranging to buy a book costing ten euros, it is possible to lose the money if the book is not delivered at all, or the result is otherwise not considered worth the investment. It is, however, rather unlikely to suffer losses higher than the ten euros invested in the action. This would mean that even though a trustee's reputation may suggest they occasionally defect in a way that causes considerably greater monetary losses than the price of a book, it would only be possible for them to defect with a smaller loss within this particular action.

As risk builds on past information expressed as reputation, it must also include a measure for the quantity and quality of the information there exists about it. We measure the amount of information available, the amount of expressed uncertainty in it, and the credibility of its sources; these measures are discussed further in Section 3.2.

3.1.3 Risk tolerance

A trustor's risk tolerance is determined by the situation calling for a trust decision. While the risk involved depends on how the trustee usually acts and what it will do in the future, risk tolerance does not depend on the trustee's behaviour. Risk tolerance encompasses both tolerance of certain probabilities of various outcomes, and of the uncertainty in the information. An example risk tolerance evaluation, applied to the risk evaluation presented earlier, is depicted in Figure 3.4. In the example, one of the constraints is not met: the probability of a slight negative effect for the reputation asset is higher than the limit of 0.1 set for it.

Risk tolerance depends on the business importance of the action, and local policy expressing the trustor's general risk attitude. It is expressed as a set of constraints for the risk evaluation; if the acceptance constraints are met, the trust decision is positive. The constraints are asset-specific, and can give upper or lower bounds either to probabilities of particular outcomes or to the sum of probabilities of a set of outcomes. The bounds can be absolute or relative, containing comparisons between probabilities: the probability of a monetary gain can be required to be larger than the probability of loss, for example.

Risk tolerance constraints can also give asset-specific bounds to the uncertainty measures attached to the risk evaluation. These are particularly useful in determining when a decision is not routine enough to be handled automatically, and instead needs to be delegated to a human user.

A trustor's risk attitude determines how risk-averse or risk-seeking the trustor is. A risk-averse trustor will require that an action have high importance to balance for the risk a positive decision would cause, while a risk-seeking trustor can accept a higher risk in relation to the baseline set by the action's importance. A trustor's risk attitude cannot be determined out of context. Actors can have different risk attitudes for different assets due to different relative valuations for them, and their attitude can change over time or be affected by very complex notions of measured utility. Ex-

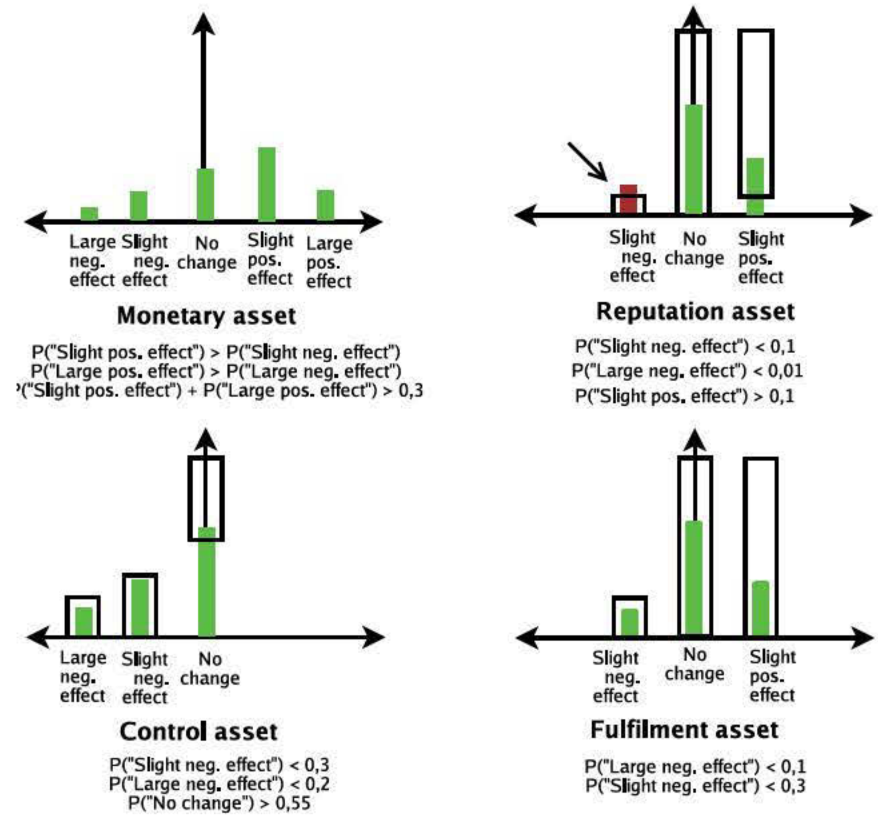

Figure 3.4: An example risk tolerance evaluation, where one risk value falls outside the constraints.

ternal pressure can shake an agent to adjust its attitude: for example, organizations threatened by bankruptcy may attempt even desperate actions to avoid it even when the gamble could lead to a considerable increase in debt.

Building a configuration system to help a trustor express their risk attitude through these formulae is an important item of future work. The aim is to bring the level of expression for configurations as close to the business processes and the language of the decision-makers as possible, and minimizing configuration work that requires expensive consultation.

3.1.4 Reputation

As risk estimates aim to define probabilities of different futures, they must examine the past to learn from it. Reputation represents information of the past, and provides a basis for risk evaluation. This step requires us to make the assumption typical to reputation systems: that trustees are, in fact, generally sufficiently consistent in their behaviour for the past to indicate anything about the future.

Reputation represents the current view of a trustor's trustworthiness formed from local experience and third party experience. The reputation views building on these two very different sources are stored separately up until the moment of a trust decision.

The reputation of a trustee is independent of the action being considered. Experiences are expressed based on their effects on different assets, not what kind of action caused them: in this utilitarian approach, the thought does not count — only the outcome matters. This approach allows us to make decisions based on specific risks, while avoiding the problem of having information become too sparse. We also do not include a separate notion of competence in performing particular types of actions [Vil05b] in our trust model: if an enterprise publishes a service offer that describes the interface to access its service and then repeatedly fails to provide it, it has either falsely claimed to be competent, or has refused to offer it. The end effect to for example our monetary and fulfilment assets is not very different either way.

Both external and local reputation views follow the same format. They build on the asset model similarly to risk, and store the number of experiences in different outcome types for each asset. A reputation view also stores the number of cases where no outcome type could be determined for the asset, which is a factor in determining the quality of the available information. To accommodate the credibility analysis of external information sources, reputation views also store the overall credibility value for the reputation view, a real number between 0 to 1 used in adjusting the overall uncertainty estimation. We will return to the relationship between local and external reputation in Section 4.

3.1.5 Importance

The importance factor expresses the business value of the action, and the cost of a negative trust decision. It represents the costs and benefits related to the action which do not depend on the behaviour of the trustee. For example, a negative trust decision blocking an action may result in compensation clauses being activated in the contract between the trustor and trustee. This is simply another factor to consider in the decision: the required compensation may still be small enough that blocking the action is preferable to the risk that the trustee causes greater losses by defection.

Importance information covers the investment required by the action and the certain return of investment, when for example a certain group of actions are considered to be so valuable that requests for them get a high priority despite the cost. For example to a bank, a cheap loan may be the strategic way to attract customers to move all their banking services to it. Importance should also capture a lack of real choice, should it occur, and more generally the perceived cost of denying service to the trustee. The valuations considered may include the interests of the eCommunity, but adjusted based on how much weight the trustor decides to place on them.

The nature of trust decisions in the TuBE system demands that importance be expressed in the form of assets, and gains or losses to each asset caused by approving the action. For a particular action and its parameters, its importance is defined as the effect it has for each asset. This basic process of generating importance values is depicted in Figure 3.5.

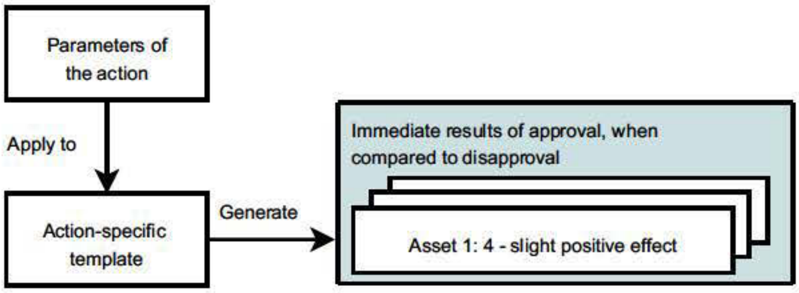
Parameters of the action
Apply to
Action-specific template
Generate
Immediate results of approval, when compared to disapproval
Asset 1: 4 - slight positive effect

Figure 3.5: The process of generating importance values.

3.1.6 Adjusting to business fluctuations through context

The aforementioned four factors of risk, risk tolerance, reputation and importance represent the valuations, expectations and policies across the enterprise and are relatively fixed in their format as a function of time. However, the business world fits poorly into rigid frameworks: the risk evaluation of a particular partner should consider that they are forced to have an insurance which makes a considerable monetary loss impossible; another partner is a valued contact, and therefore it is particularly important to collaborate with it even in the face of some additional risk. A third partner's reputation may suddenly plummeting due to what is suspected to be a misinterpretation.

Both the environment and the internal policies of an enterprise are in constant fluctuation. A trust management system for inter-enterprise collaborations must be prepared to handle frequent, often temporary adjustments to what is otherwise a clear set of policies and valuations. In order to retain clarity in modelling while catering for the messy reality, we have opted to add a fifth element to trust decisions: context filters.

Context filters are not a value factor by themselves. Instead, they adjust the four factors described above to temporary changes and special cases. This kind of context is applied as modification filters whenever a factor is evaluated for a trust decision. Each context filter works on a particular factor, and when its triggering constraints are met, the modification it defines is applied before the value of the factor is passed onwards in the decision-making. A modification could for example involve moving the probability of a considerable monetary loss to add to the category of minor monetary loss, to express the effect of an insurance with an excess.

3.2 Making multi-dimensional trust decisions

Trust decisions are made on joining a collaboration and relevant points during it, whenever new resources are committed. They are computed locally by the policy-enforcing monitor within the business service. A decision is situational and has a limited scope: new decisions are needed regularly to take new experiences and other changes in the background situation into account. A level of distrust is always present, as even after a positive trust decision, the behaviour of collaboration partners is monitored to catch any violations of the collaboration contract or shared policies.

A trust decision is formed from two tracks: on one track, building the risk estimate of the situation, and on the other, building a measure for risk tolerance. The risk estimate is built from the reputation factor, and risk tolerance from the importance factor. All four elements can be adjusted by context filters before their evaluation at the next step, which changes their values but not structure. We omit the context adjustments here in favour of a clearer description of the central processes.

A represents the set of guarded assets, represented by integers on the range $0..|A|$. Assuming that the standard assets are used, $|A| = 4$. J represents the set of possible outcomes as integers between 0 and 5: 0 for unknown effect, 1 for major negative effect, 2 for minor negative effect, 3 for no effect, 4 for minor positive effect, and 5 for major positive effect. $|J| = 6$. "No effect" differs from "unknown effect". For example, not losing or gaining money would represent a lack of effect, while an experience with a delayed payment on its way might be included as an unknown outcome in decision-making.

The risk R of an action contains $|A|$ vectors $\mathbf{r}_a$, one for each asset:

$$\mathrm{R} = (\mathbf{r}_0, \mathbf{r}_1, ..., \mathbf{r}_{|A|-1}), \text{ where } \mathbf{r}_a = (\mathbf{p}_a, |E|, c, q_a)$$

The vectors store the probabilities of different known outcomes for each asset, and three different measures of the amount and quality of the information used to produce the evaluation. The risk evaluation is specific to a given trustor, trustee and action; we omit representations of these three parameters in the formalism for readability.

The first term $\mathbf{p}_i$ represents the probabilities of different outcomes of the action: $\mathbf{p}_a = (p_{a,1}, ..., p_{a,|J|-1})$, where each $p_{a,j}$ is the calculated probability of outcome j happening for asset a. The latter indexes begin from 1; the probability of an unknown outcome is not considered, but information about unknown outcomes in the past is represented through another term, q_a. We require that the probabilities add up to 1 for all assets, i.e. $\forall a \in A : \sum_{j=1}^{|J|-1} p_{a,j} = 1$.

E represents the group of all experiences the trustor has on the trustee in its reputation view, and the number of experiences, $|E|$, measures the amount of reputation information behind the risk estimation. The third term, c, is the combined credibility of local reputation and external, third-party reputation information at the time of evaluation. The last measure, q_a, is the number of experiences in E where the outcome was unknown for asset a: the higher this value is in relation to $|E|$, the lower the certainty of the risk analysis. We return to these quantity and quality measures when discussing the transformation of reputation information into a risk analysis.

The reputation U of a trustee as viewed by a given trustor consists of two halves: a local reputation U^{local}, and a local evaluation of third-party reputation information U^{ext}. This reputation is by no means global: it is not agreed upon by the members of the reputation network, and it is trustor-dependent like local reputation. As discussed in the previous section, reputation is not action-dependent: only the observed effects to assets are measured, and it does not matter what action caused them.

The structure of both the local and external reputation information is the same: each contains $|A|$ vectors and a credibility score c^{local} or c^{ext} in the range [0..1]. The credibility value c^{local} of local reputation is 1, while the third-party reputation credibility value c^{ext} is set based on a local analysis of the combined credibility of the reputation system the information comes from, and the credibility of the sources providing reputation information in that network. To avoid repetition, we present the two symmetric reputation structures as generic variables that are equal for both halves, denoting this with an asterisk (*). For example, U^* represents both U^{local} and U^{ext}, with the exact type left undetermined.

Both types of reputation consist of $|A|$ vectors $\mathbf{u}_a^*$, one for each asset:

$$\mathrm{U}^* = (\mathbf{u}_0^*, \mathbf{u}_1^*, ..., \mathbf{u}_{|A|-1}^*)$$

Each vector $\mathbf{u}_a^*$ consists of six counters $u_{a,j}^*$, which express the number of experiences of outcome $j \in J$, where J, as mentioned earlier, is the set of outcome categories $\{0, 1, 2, 3, 4, 5\}$, with $j = 0$ representing an unknown effect. Each counter $u_{a,j}^*$ represents the number of experiences where the outcome for the asset a was as indicated by the value of j.

The set of experiences, E^*, is a group

$$E^* = \{\mathbf{e}_k^* : e_{k,a}^* = \text{ outcome value } j \in J, \forall a \in A\}.$$

In other words, an experience consists of the effects of one action expressed for each asset.

Given the set of experiences, the reputation counters u_a^*, j can be expressed as as the size of the subgroup of E where the experiences had outcome j for asset i, that is: $u_{a,j}^* = |E_{a,j}^*|$, where $E_{a,j}^* = \{\mathbf{e}_k^* \in E^* : e_{k,a}^* = j\}$. The value $u_{a,0}^*$ is is particularly interesting, as it expresses the number of experiences with unknown values for the asset. When compared to the total number of experiences, $|E^*|$, it provides us a measure on the quality of the information.

Reputation is transformed into risk by 1) merging the local and external reputation views together with a weighted sum, 2) scaling the experience counters representing known effects to proportions in the range [0..1] and applying an action-specific *base risk* to trim impossible outcomes, and 3) recalculating a joint credibility and information content score for the result, that is, the variables c, $|E|$ and q_a that appear in the risk vectors.

The local and external reputation views are merged based on the amount of information available in either, and the credibility attached to the views. Local reputation is more credible than external, but there is usually less local information available, and both should be reflected on the weight given to local reputation. We define two functions, μ^{local} and μ^{ext}, to determine the weights for both local and external reputation values. They use the corresponding credibility value c^*, amount of experience $|E^*|$ and a vector $\mathbf{q}^*$ of the number of experiences where the effects are unknown for different assets: $\mathbf{q}^* = (u_{0,0}^*, u_{1,0}^*, ..., u_{(|A|-1),0}^*)$. The multipliers produced by the μ^* functions add up to 1; the specific declaration of the μ^* functions depends on local calibration. In other words,

$$\mu^{local}(c^{local}, |E^{local}|, \mathbf{q}^{local}) + \mu^{ext}(c^{ext}, |E^{ext}|, \mathbf{q}^{ext}) = 1.$$

The merging formulae can be generalized to handle more than the two categories of reputation information as well, as long as the weights defined by the μ^* functions add up to 1. We present an extension to the base information model taking advantage of this generalization in Section 4.1.5: We divide both local and external reputation information into multiple reputation epochs, which capture clearly different periods of trustee behaviour. This division allows us to give more weight to the newest subset of experiences, and react to changes in behaviour more quickly. The extension provides a new adjustment point to the information model and decision-making algorithm.

Using the μ^* functions, we merge the experiences into a temporary $\mathrm{U}^{merged} = (\mathbf{u}_0^{merged}, \mathbf{u}_1^{merged}, ..., \mathbf{u}_{|A|-1}^{merged})$, where each vector $\mathbf{u}_a^{merged}$ contains six combined counters: the weighted sum of the local and external respective counters. Note that unlike the values in $\mathbf{u}_a^*$, these merged values are no longer integers, but real numbers. For all $a \in A, j \in J$, we have:

$$u_{a,j}^{merged} = \sum_{* \in \{local, ext\}} \mu^*(c^*, |E^*|, \mathbf{q}^*) * u_a^*, j$$

In the second phase, we scale the experience counters except the unknowns to the range [0..1] to represent their relative frequency and future probability. To achieve this, we sum the values of known effects ($j \neq 0$) and divide each value by the sum. As a result, we get a set of action-independent values $p_{a,j}^{generic}$, corresponding to the values mentioned earlier in the risk representation. These values differ from the final $p_{a,j}$ values in that they do not include the notion of the base risk of the action yet.

$$\forall j \in J \backslash \{0\} \; p_{a,j}^{generic} = \frac{u_{a,j}^{merged}}{\sum_{j=1}^{|J|} u_{a,j}^{merged}}$$

As some actions can in practice never cause all outcomes, we use the concept of base risk to merge the probabilities of outcomes impossible for the given action into the probabilities of possible outcomes, to produce the final $p_{a,j}$ values. Base risk, $\beta_a(\mathbf{p}_a^{generic})$, is an action-specific function defining how this merge is done. An action with little influence, for example, should have the probabilities of major negative and positive outcomes added to the corresponding minor outcomes and set to 0: given $\mathbf{p}_1^{generic} = (0.1, 0.2, 0.5, 0.2, 0.1)$ we would then have $\mathbf{p}_1 = \beta_1(\mathbf{p}_1^{generic}) = (0, 0.3, 0.5, 0.3, 0)$.

In the third phase, we calculate combined measures of the quality of information: c, $|E|$, and the $|A|$ different q_a values. The combined credibility c is determined by a μ-weighted average of the local and external credibilities. It depicts the weight given to each half in the probabilities as well.

$$c = \sum_{* \in \{local, ext\}} \mu^*(c^*, |E^*|, \mathbf{q}^*) * c^*$$

To calculate the total number of experiences, $|E|$, we add together the number of local and external experiences: $|E| = |E^{local}| + |E^{ext}|$. Although it is clear that not all experiences have been given equal weight in the evaluation, this measure gives an indication of how much information there is available on the actor overall. The combined number of experiences for each asset where the effect was unknown, q_a, is gotten by adding the values of the previously calculated $\mathbf{q}^*$ vectors, for all $a \in A$:

$$q_a = q_a^{local} + q_a^{ext}$$

Again, not all unknowns were weighed in the probability calculations equally, so we could consider a μ-weighted average here similarly to the calculation of the credibility value c as well. On the other hand, the true total number of unknowns is a more useful value to use together with the amount of total experience, $|E|$, as $q_a/|E|$ gives the proportion of uncertain values in the experience set.

The risk evaluation must be compared to the risk tolerance to produce a trust decision. The risk tolerance T of an action, given a particular trustor and trustee, consists of two vectors of $|A|$ functions $f_{x,a}$, one for each asset. The first vector represents the value bounds for automatically accepting an action, while the second vector represents the value bounds for automatically rejecting an action.

$$\mathrm{T} = (f_{accept,0}, f_{accept,1}, ..., f_{accept,|A|-1}), (f_{reject,0}, f_{reject,1}, ..., f_{reject,|A|-1})$$

The functions represent the acceptable limits for the risk values in the risk vectors $\mathbf{r}_a$: they evaluate whether the values are within bounds or not. If the values do not fit fully within automatic acceptance or rejection, the decision is delegated to a human user.

$$\forall a \in A, f_{x,a}(\mathbf{r}_a) = \begin{cases} 1 \text{ if the values of } \mathbf{r}_a \text{ are within the bounds} \\ 0 \text{ otherwise.} \end{cases}$$

Risk tolerance is a function of the importance of an action. The importance factor I contains $|A|$ values v_a, one for each asset.

$$\mathrm{I} = (v_0, v_1, ..., v_{|A|-1})$$

The values express the known effects a positive trust decision has on different assets: $\forall a \in A : v_a =$ an effect value $j \in J \backslash \{0\}$. There are no unknown effects ($j = 0$) in importance: it depicts only those assumed effects and valuations in the enterprise that affect decision-making.

Both importance and risk tolerance depend on the trustor, trustee and action. Risk tolerance is evaluated based on the importance value; each trustor determines the exact evaluation function $\Phi_{\mathrm{T}}(\mathrm{I})$ that produces the risk tolerance T.

When a trust decision is needed, the two computation tracks described above are executed. With a risk evaluation generated from reputation values, and risk tolerance functions derived from importance values, the actual trust decision is straightforward. An example calculation based on local experiences and no adjustments from base risk formulae is depicted in Figure 3.6; the probability of "loss" is shorthand for the sum of probabilities of negative outcomes, and "gain" of positive outcomes.

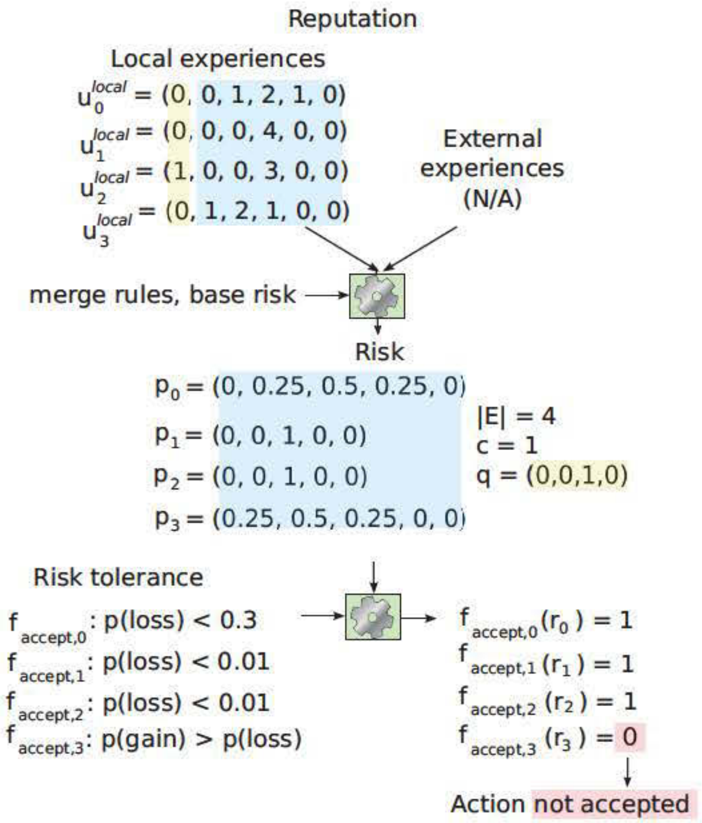
Reputation
Local experiences
$u_0^{local} = (0, 0, 1, 2, 1, 0)$
$u_1^{local} = (0, 0, 0, 4, 0, 0)$
$u_2^{local} = (1, 0, 0, 3, 0, 0)$
$u_3^{local} = (0, 1, 2, 1, 0, 0)$
External experiences (N/A)
merge rules, base risk
Risk
$p_0 = (0, 0.25, 0.5, 0.25, 0)$
$p_1 = (0, 0, 1, 0, 0)$
$p_2 = (0, 0, 1, 0, 0)$
$p_3 = (0.25, 0.5, 0.25, 0, 0)$
$|E| = 4$
$c = 1$
$q = (0,0,1,0)$
Risk tolerance
$f_{accept,0}$: p(loss) < 0.3
$f_{accept,1}$: p(loss) < 0.01
$f_{accept,2}$: p(loss) < 0.01
$f_{accept,3}$: p(gain) > p(loss)
$f_{accept,0}(r_0) = 1$
$f_{accept,1}(r_1) = 1$
$f_{accept,2}(r_2) = 1$
$f_{accept,3}(r_3) = 0$
Action not accepted

Figure 3.6: An example trust decision calculation.

The evaluation result of a vector $\mathbf{f}_x$ is a match if the risk tolerance functions within it evaluate to 1, i.e. $f_{x,a}(\mathbf{r}_a) = 1$ for all assets $a \in A$, and a mismatch otherwise. The action in turn is automatically accepted if the first vector $\mathbf{f}_{accept}$ is matched, automatically rejected if the second vector $\mathbf{f}_{reject}$ is matched, and delegated to a human user otherwise. The division of the risk domain into separate areas for positive ($\mathbf{f}_{accept}$), negative ($\mathbf{f}_{reject}$) and uncertain trust decisions is depicted in Figure 3.7. Multiple function vectors $\mathbf{f}_x$ can be used to generate a set of tolerance constraints for more detailed decisions, for example to separate an additional category for "automatically accept, but with increased monitoring".

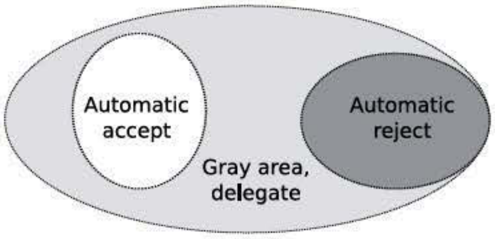

Figure 3.7: Tolerance functions divide the risk value domain into three areas.

3.3 Related work on modelling trust decisions

This section presents examples of work on supporting the expression of trustor-specific trust decision policies. We will compare the TuBE information model to another risk-aware trust model, designed for pervasive environments, and discuss existing state-of-the-art policy languages and their suitability for expressing the policies in TuBE. We will discuss systems that have their main focus in reputation management and the organization of reputation networks later, in Section 4.3. For additional background, we have published three surveys: the first survey on trust management research focuses on the differences of trust management models [RK05]; the second studies different factors used in trust decisions [Vil05b], and the third survey analyzes credibility evaluation support in different reputation systems [RKK07].

3.3.1 SECURE — a risk-aware trust model

SECURE (*Secure Environments for Collaboration among Ubiquitous Roaming Entities*) is an international project within the European Commission

Fifth Framework Programme. It aims to provide a framework for trust management in ad hoc environments [C$^+$03]. Example application areas include a user determining whether to give electronic money to an unknown vending machine, or to play poker in a network of strangers. In both cases, a small personal device acts as a trust decision support or an automated decision maker for a private person, using the calculated risk for the action as a basis.

The environment sets a requirement for the trust management system to be able to operate on uncertain information. Identity management becomes an essential part of the system, because there is little or no infrastructure for strong identification to rely on. In this sense, the challenges for SECURE differ from those in inter-enterprise collaboration, where the infrastructure is required for negotiating contracts. Identity management and entity recognition solutions in ad hoc environments are a wide problem area [C$^+$03, SFJ$^+$03] that falls outside the scope of this thesis.

The basic trust decision model of SECURE uses information about the trustee's past behaviour as a basis to calculate a risk estimate of an action. Risks are represented as probability distributions of different costs and benefits [C$^+$03, EWN$^+$03]. An example composition of a cost-benefit probability distribution function is presented in Figure 3.8.

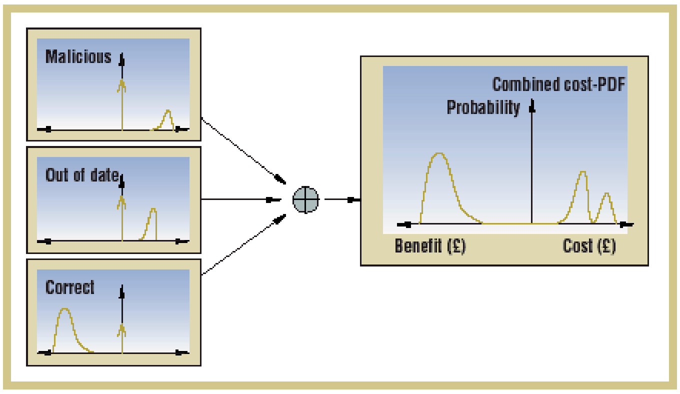

Figure 3.8: Combining the probability distributions of different outcome types [C$^+$03].

Experience information is used to select a specific probability distribution for each outcome [C$^+$03]. The information can be a combination of external reputation and local observations; the two types of experience are kept semantically separate in the model [EWN$^+$03, WCE$^+$03].

SECURE includes two central features in the context of trust management models: First, it explicitly models uncertainty in the information leading to a trust decision, which is a necessity for systems relying on unreliable information sources. Second, it uses a risk model which is aware of the stake in a trust decision: the cost-benefit probability distribution functions are able to differentiate between major monetary losses and minor monetary losses as well as the probability of loss as opposed to the probability of gaining something.

The SECURE risk model differs somewhat from the model in TuBE. Where TuBE has a discrete, five-step outcome scale for four different assets, SECURE has one continuous scale of costs and benefits in terms of a single asset type. In TuBE, we have chosen a discrete risk scale in the name of simplicity, while allowing more than one asset type to avoid the need to convert clearly different types of stakes into a single value range. Due to the the aim to automatically process more complex, inter-enterprise collaboration situations, TuBE also has somewhat different information needs from SECURE, including the business importance of proceeding with the action and context filters to adjust the input.

SECURE provides a versatile information model to support different trust decision policies and brings out some requirements it has for a policy language. Like TuBE, it focuses on building an overall framework, and does not set out to specify its own trust management policy language. In the next section, we look into work on specifying policy languages for trust management, and evaluate how well the state of the art in policy languages matches the needs of our trust information model and process.

3.3.2 Policy languages for trust management

Trust management policy languages can be roughly divided into two categories: the traditional, certification-based approaches — which also first adopted the term "trust management" to denote their emerging branch of access control solutions — and the new approaches, which are based on reputation information and experience. The two directions address different levels of decision-making: certificates and delegation rules can be used to determine whether a service request is connected to an existing contract, while an experience-aware policy can determine whether the contractually authorized partner is currently sufficiently trusted that they can access the service on one hand, and resources should be invested in the collaboration with their service on the other hand. In the case of a negative decision, other options include making a possible contract violation in order to avoid risking further resources, or for example introducing additional monitoring to keep an eye out for further suspicious activity. These two levels have also been referred to as hard and soft security [RJ96], referring to the automated learning and dynamic adjustment enabled by the experience-based systems.

The certification-based approach to trust management has its roots in flexible access policies proposed by Blaze, Feigenbaum and others. The three most well-known systems in the field are PolicyMaker [BFL96], KeyNote [BFK98] and REFEREE [CFL$^+$97]. In the Web Services context, the approach is also adopted by the Web Services Trust Language WS-Trust [OAS07].

The evaluation of a certification-based policy is based on the exchange of cryptographically signed certificates or security tokens, which can be used to express authorization to access a specific service directly, or more abstract notions such as identity, group membership or delegation of authority. These kinds of general-purpose certificates can be used as a basis for access policies such as "trustee can present sufficient proof that it is a business service of Partner A, or of a subcontractor whom A has authorized to operate in its stead for this purpose". This way, it is not necessary for the trustor to produce new authorization certificates every time Partner A decides to authorize a different subcontractor, nor does the trustor even need to know who exactly is subcontracting for A at the moment.

As discussed in Section 2.3.1, the TrustCoM [W$^+$06] framework for trust, security and context management represents this certification-based approach. While it does have a reputation service, it is only used for finding new partners, while the certificate-based approach is used for access control during a collaboration.

In the experience-aware approach, reputation information can be connected into the trust decision policy. Few policy languages include the notion of reputation or experience, however. As such, they are not directly applicable to the needs of TuBE.

The IETF RFC 2753 defines requirements and architecture options for a general framework for policy-based admission control based on policy enforcement points (PEP), typically on a network perimeter, and policy decision points (PDP) which are consulted by the former to reach a decision [YIP$^+$00]. The contents, input or format of the policies themselves are not specified, although recommendations are given on e.g. ensuring that conflicting policies can be ordered based on priority to determine which one is followed.

Grandison and Sloman's SULTAN (Simple Universal Logic-oriented Trust Analysis Notation [GS02, GS01]) has been one of the first proposals for this kind of trust specification. It builds on the Ponder policy specification language by Damianou et al. [DDLS01]. SULTAN has focused on analyzing trust relationships: while it has a notion of a context for trust, it is not risk-aware, nor does it consider incentives to trust beyond reputation or experience information.

The SULTAN policies specify situations and conditions for specific levels of trust or distrust, and willingness to recommend a trustee positively or negatively [GS01]. For example, we can specify that Alice trusts Bob at level 50 (on a scale from 1 to 100) to perform the action of transferring money, if Bob's bank is considered secure:

trust(Alice, Bob, moneyTransfer(Amount), 50) ? bankSecure(Bob);

The conditions may contain references to other policies. For example, Alice may trust Bob only if Cecilia recommends him for this action at a high enough level.

In SULTAN, changes in both trust and recommendation levels are expressed as new policy entries: beyond the policy set, the notions of trusting and recommending, or experiences and recommendations do not exist outside the policy database: the data and policy are combined as policies about the outcome of the data. In TuBE, reputation information exists independently in a repository, and decision policies use it and risk information calculated from it as parameters in order to determine whether an action is allowed. The same data can produce different results with different policies, which sets TuBE apart from the approach adopted by SULTAN.

The ENFORCE project proposes a framework for specifying and analyzing trust management policies [LMS+07]. Based on a case study, Solhaug et al. argue that the graphical UML sequence diagrams are suitable for policy expression [SES07]: using a graphical, but formally defined language makes the policies more user-friendly to for example management representatives, while still providing support for automated processing. In Deontic STAIRS, the UML sequence diagram profile is extended to provide a means for specifying trust management policies [RSS08]. An example trust policy is presented in Figure 3.9.

The example resembles an expression of risk tolerance policy in TuBE. The "est.p" measure represents the subjective probability that the customer will pay back a granted loan. In other words, it is a single-dimensional risk estimate. This level of expression is insufficient to cover our own multi-dimensional trust information model; on the other hand, it expresses the triggers and outcome of trust decisions, which are beyond the immediate scope of this work and yet necessary for operating TuBE. The question remains how well Deontic STAIRS is in practice suited for this purpose in

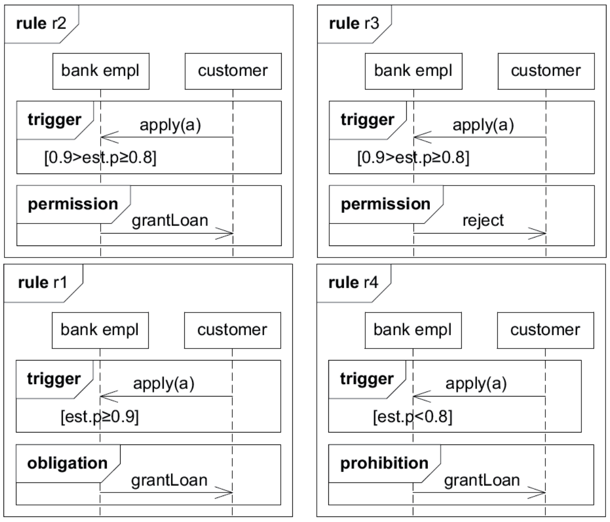

Figure 3.9: A trust policy expressed using Deontic STAIRS [RSS08].

automated inter-enterprise collaboration management; this will have to be evaluated separately.

The rules for obligation, prohibition and permission to grant a loan correspond to TuBE's automated accept, automated reject, and the gray area in between that must be forwarded to a human user to decide. On the gray area in the example, for probabilities between 0.8 and 0.9, the policy simultaneously allows the bank employee to grant or to reject a loan. If the probability is below 0.8, the employee is prohibited from granting the loan, and if it is above 0.9, she is obliged to grant it. This property of deontic logic is very welcome for supporting semi-automated decision-making. If we limit the scope of what we expect the policy to cover, we can work around the limitations of the risk model by considering the statement "est.p < 0.8" in the examples to be equivalent to "the trust management system return value is automatic reject", in which case the monitor enforcing the Deontic STAIRS policy does not need to be aware of what kind of information model TuBE uses to produce this result.

In addition to using graphical policy languages, a further usability improvement would be to enable policy-setters to express policies using business concepts, independently of the technical implementation used to enforce these policies. In his thesis, Solhaug discusses the notion of policy refinement [Sol09]: the process of making a high-level policy specification more concrete, and bringing it closer to implementation and enforcement.

This idea is similar to the more generic concept of vertical model refinement in model-driven engineering (MDE) [Sch06, MG06]. While the initial process of mapping business concepts to technical concepts must be done manually, the propagation of changes to high-level policy to the more refined policies closer to implementation can be automated, once the transformations are set. Further developments in policy refinement would also be highly useful from the point of view of Pilarcos and TuBE, as a central goal of our work is to bring policy-setting to the level of business concepts, while the implementing rules enforced by monitors must by necessity be very simple and straightforward to be efficient to process.

In expressing risk, ENFORCE builds on the CORAS conceptual framework for security risk analysis; Stølen et al. have earlier proposed a UML profile for the risk modelling language [LdBSV04, BS04]. The group has extended the risk work further with support for modelling how different decision policies influence the analyzed risks [RSS08].

The CORAS modelling methodology for risk analysis has a relevance beyond providing a basis for ENFORCE: it allows us to identify action-specific differences in risk. The business network models discussed in Section 2.1 can be annotated with risk information as they are produced by domain experts; the enterprises using the models can then decide whether they wish to take the information into account in their trust decisions.

In comparison to the policy examples presented here, TuBE policies are quite close to the implementation level, as they operate directly on the trust information stored in the system. For this reason, the language for expressing the policies will be close to a generic programming language. All TuBE policies can be seen to follow a common format of conditions and effects: if the conditions are met, the system should perform the following actions.

For example, a risk tolerance policy of the form “if there is a risk of high monetary loss, reject the action” could correspond to an expression that states that within the risk value vector R, if the probability $p_{0,0}$ of a major negative effect on the monetary asset is higher than 0.1, send a response to reject the action, otherwise accept it.

The conditions and the effects both manipulate parameters that are the factors from the information model, such as the action identifier, or the source of a piece of reputation information. The conditions examine these parameters and decide whether the inputs warrant triggering an effect, which then either manipulates the parameters or sends a message to an external agent stating whether the action should be accepted, rejected or forwarded to a human user. These requirements make policy definition resemble scripting the trust management system, and it is therefore relatively uninteresting to define a policy language specifically for TuBE.

Policy refinement makes it possible to configure TuBE using higher-level policy languages. These can then be automatically mapped to the policy languages that TuBE enforces. The user interface for configuring the trust management system can abstract away the different TuBE policies into a set of trust profiles, to further simplify the configuration task. The expressiveness of the information model and the policy points in the decision algorithm support a broad range of decision policies, while the details of designing good user interfaces for configuring the system are outside the scope of this research.

3.4 Chapter summary

In this chapter, we have presented an information model for computational trust decisions, and the algorithms for translating the gathered information to trust decisions. The information model distinguishes between different assets and different impacts that actions can have on them, ranging from major negative to major positive effects, which makes the model more expressive than the state of the art.

The expressive power in the different factors of the model forms the basis for flexible decision policies, which allow the trust management system to adjust to different decision contexts. The expressive power also serves to make the system less vulnerable, as many attacks are based on the limitations of the trust information model. If the model is not aware of transaction value, for example, it is vulnerable against an attacker gaining good reputation through low-value transactions and then misbehaving in high-value transactions to cash in its reputation. The decision-making algorithm remains reasonably straightforward and scalable despite the necessary tradeoff for expressive power. Multiple measures for the quality of the available information are carried through the process to ensure that they are available at the decision time, so that decision policies can also set limits to when the information is not reliable enough to make an automated decision.

We have presented relevant trust management policy language efforts, and found that while their underlying trust information models are too narrow for use in TuBE decision policies, the work on Deontic STAIRS and the underlying CORAS modelling methodology show promise in the context of business network modelling as well as configuring the triggers and outcomes of trust decisions. We identify policy refinement from the business level to the implementing level as an interesting branch of future work as well. As the implementation-level policies of TuBE resemble scripts for the trust management system, we have chosen to not set out to explicitly define a TuBE-specific policy language in this work.

Chapter 4

From reputation to trust decisions

This chapter focuses on reputation management, opening the third and final core area of research contained in this thesis. In order for the TuBE trust management system to learn from experience, it must aggregate reputation information from different sources. In this chapter, we first describe the aggregation algorithm for reputation: receiving, interpreting and consuming reputation information and evaluating its credibility in the first section. We then specify ways to adjust the trust management process to different business situations through configurable policies in the second section. In the third section, we look at related work in reputation management, studying common patterns in reputation systems and notable differences between the proposals. The fourth section analyzes means of achieving interoperability between reputation systems, building on the features of TuBE that allow it to connect to a number of different reputation systems for collecting experience information needed for trust decisions.

4.1 Learning from reputation information

In TuBE, the reputation of an agent consists of two types of information: local reputation, based on first-hand experiences, and external reputation gained from other actors through reputation networks. They are collected and stored separately, using different updating processes, and only combined at the point of making a decision.

Keeping the two types of information separate has three benefits. First, it makes a clear division between information produced by the local system that can be shared in external reputation networks, and information originating from the networks. External information should not be uncritically fed back into the network in order to avoid amplifying single opinions by repetition [JMP06]. Second, during decision time, it allows us to change the weighing between the two types independently of when or how the information has been acquired. Third, it allows us to compare external and local reputation in order to evaluate the quality of external reputation sources.

In this section, we discuss the algorithms for receiving, interpreting, evaluating the credibility of and consuming experience information both locally and through reputation systems. We present how local observations in the form of monitor translate to experiences, and how external reputation flows are connected into the system.

We also present three extensions to the basic process of producing and consuming reputation information: the local credibility evaluation of reputation sources and the use of credibility information in decision-making, a reputation system based on objective and verifiable experiences, and reputation epochs for detecting and reacting to changes in reputation flows. Of the above three aspects, we have chosen to focus on the credibility information in the simulation experiments presented in Chapter 6, as it connects to the entire trust management process, and the other two aspects depend on it. The proposed objective reputation system and reputation epochs are summarized in this section, with further evaluation presented in separate publications [RK11, RHK11].

4.1.1 Storing direct observations as local reputation

Local reputation represents the first-hand experiences of the trustor, and the information is considered more valuable and semantically clear than information from third parties. As a downside, as gaining first-hand experiences requires taking the risk of collaborating, there is usually less local reputation information available. It also does not help making decisions on new potential partners, on whom there are no earlier first-hand experiences. The main value of local reputation is in reliably detecting behavioural changes, and validating the quality of external reputation sources. In addition, local experiences are used when participating in the operation of collaborative reputation systems and providing new information to them.

Local reputation is built by consolidating single experiences. One experience item describes the outcome of one action, and it stores the outcome of the action from the point of view of each asset. If it is unclear what the outcome of the action was for an asset difficult to monitor, a special "no outcome value" is recorded for that asset instead, representing an unknown impact.

Monitor rules must be set up so that a group of monitor events can always be either formed into a single outcome or into the undetermined non-value. No separate values are given for specific undeterminations, such as "the outcome is either x or y, but not z"; we assume that when an outcome is undetermined, it will either become fully determined when more information arrives, or remain fully uncertain due inherent limitations in the monitoring facilities. The experience item consolidation process is shown in Figure 4.1.

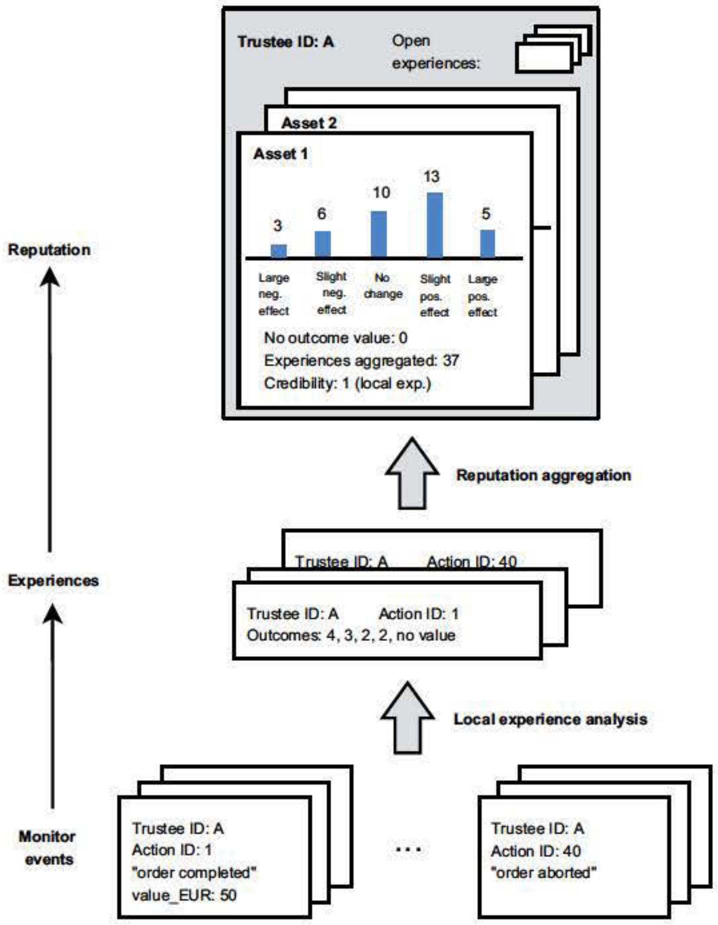

Figure 4.1: The process of transforming monitor events into a local reputation view.

Experiences are formed by analyzing the output of the Pilarcos monitors [KMR05]. These monitors are not aware of the particular assets being protected in the system, but only detect configured noteworthy events, such as the start and end of a financial transaction. These events are represented by a locally unique type identifier and possible simple parameters. They must be set to the context of a given action and trustee, as is experience.

An event could be e.g. “product order”, with the price or value of the product as a parameter. Further parameters would be the identifiers of the trustee and the action whose message traffic caused the event.

A specialized Pilarcos monitor plugin focuses on collecting experience information specifically; it is not concerned with e.g. general intrusion detection. Other kinds of monitoring can detect unknown actors sending inappropriate requests, denial of service attacks or other security-relevant events, but as these cannot be connected to a specific contractual context, they are not usable for experience management. They can instead cause a need to activate e.g. context filters to make the system particularly careful about tying up computational resources.

The connection between Pilarcos monitors and TuBE is discussed further in Chapter 5, which presents the system architecture in more detail.

4.1.2 Gathering outsider opinions into external reputation

External reputation is formed within a set of reputation networks, each of which is an independent source of external reputation information. As noted earlier, a reputation network is a combination of its users and the underlying reputation system, i.e. the information model and the algorithms to collect, update and distribute reputation information. A single reputation system can be deployed in multiple domains, for example, resulting in separate reputation networks based on the same system.

While the reputation systems that are being currently researched are predominantly more or less distributed, we include in our definition also fully centralized, single-organization information sources, such as credit rating companies [Sta10], accreditors [Bet10, Tru10] and blacklists published by consumer protection authorities [Nat10]. We foresee that centralized information sources will continue to play an important role for reputation management between enterprises in addition to distributed reputation systems. We will return to related work on reputation management in Sections 4.3 and 4.4.

TuBE is designed to be able to connect to multiple reputation networks for gathering information. The local trustor has a representative agent in each reputation network to participate in the network, and in the case of collaborative reputation systems, feed local experience information to it. The representative agents report a trustee’s reputation in the native format of their represented reputation system, and pass it to the TuBE external reputation analysis component. The reports are transformed to experiences of the TuBE system format, and they are assigned a credibility value. This setting is depicted in Figure 4.2.

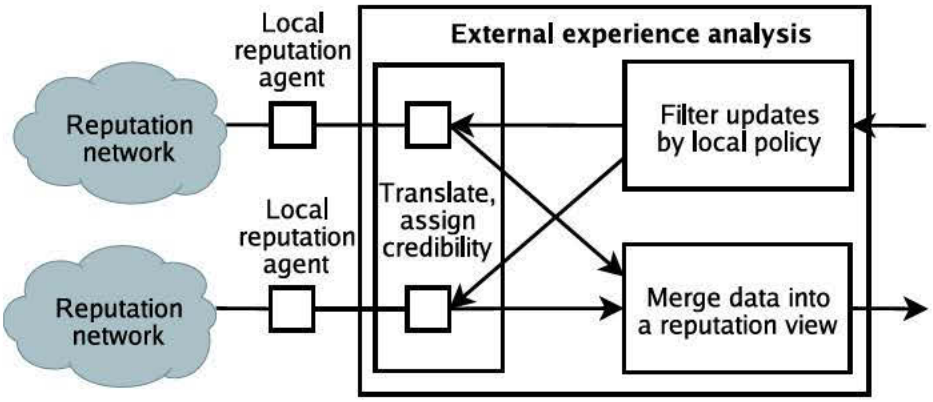
Local reputation agent
Local reputation agent
Reputation network
Reputation network
External experience analysis
Translate, assign credibility
Filter updates by local policy
Merge data into a reputation view

Figure 4.2: The experience analyzer component connects to reputation networks through local representative agents.

The views of multiple agents, as well as multiple reputation networks, must be combined to produce a unified external reputation view. The information being combined may include anything from somewhat agreeing statements to fully contradicting ones. Dealing with disagreements is a matter of policy, and has to remain adjustable by the trustor organization.

The combination can be done by several methods, such as directly preferring certain sources above others when they have a particular part of the information available, or using some kind of a combination formula such as weighted sums to produce the result. Combining information from multiple different and disagreeing sources is a broad research topic with a central role in reputation management research [RKK07, GOGGF10, JMP06]; popular approaches include only including recommendations from a trusted social neighbourhood of the actor collecting the information [GOGH08] or giving each statement a weight according to its credibility [HJS06b], as discussed in the next section. We return to this topic in Section 4.4.

4.1.3 Evaluating the credibility of reputation sources

Credibility is a form of information source "trustworthiness": its expected ability to provide correct and relevant information. In TuBE, external reputation information is assigned a credibility value between 0 and 1 to indicate how strongly the trustor subjectively believes the source to be accurate and useful. The credibility value directly influences whether the new information is consolidated into the TuBE experience storage to begin with. In addition, the combined credibility of the sources having contributed to the external reputation view of a trustee is stored with the view, again as a value between 0 and 1. The view credibility value is updated based on a changeable policy when new experiences arrive, and it is made accessible to decision-time policies as a parameter. Low-credibility information may then be given less weight in relation to local experiences in a trust decision, and it can influence the certainty measure attributed to the decision itself.

The credibility value only applies to the sources that the information was gathered from. The actual content of the item does not affect its credibility value: in other words, the analysis of reputation information by the properties of each information item itself is separated from source credibility analysis. For example, checking whether the experience differs greatly from earlier experiences from different sources does not directly support rejecting the experience as not credible; we do not know if it implies that the actor's behaviour has actually changed after earlier observations, or if the source is lying.

The source credibility rating for external reputation information items is based on a combination of two separate measurements of credibility: first, the representative agent's view of the credibility of the information based on metadata provided by the reputation network, and second, the general credibility of the reputation network as it is perceived within the TuBE system.

The reputation system underlying an external reputation network may not support assigning credibility values at all, or it may mix credibility values conceptually with, for example, the confidence of the source, in case of uncertain observations, or its certainty, i.e. the amount and quality of information behind the report. For this reason, the credibility value the representative agent assigns itself may either not be defined at all or cannot be directly interpreted as a credibility value. Credibility values from different reputation networks will inevitably have different semantics, and this must be considered when translating the reputation information into the TuBE format.

If a reputation system has a good representation of the source's own perception of the quality of the information it provides, and it supports the evaluation of the credibility of different sources well, reputation networks using the system will have a higher credibility value. On the other hand, a reputation network dominated by unreliable or malicious actors can also be assigned a lower reputation network credibility than another reputation network using the same reputation system.

The representative agent within a reputation network may attribute a low credibility value with the information item, based on receiving it from suspicious agents in the network, before it passes it to the TuBE experience analysis. The network itself may retain a relatively high credibility value within the TuBE system, because it allows agents to reason about the level of suspiciousness of other recommender agents in the network, and as such provides higher-quality information with usable credibility metadata. For the network credibility in general, a long-standing credit rating company might have a very high credibility rating, while an eBay-style open reputation network would tend to have a low credibility rating that depicts how vulnerable the underlying reputation system is to reputation attacks.

Agent-assigned and network credibility measures are combined into a single value when incorporating the reputation information into TuBE. One straightforward way of combining the agent-assigned credibility and the network's overall credibility is by a multiplication, as depicted in Figure 4.3. The credibility measures are semantically separate, and other combinations are also possible. The combination takes place when experiences are incorporated into the local reputation store.

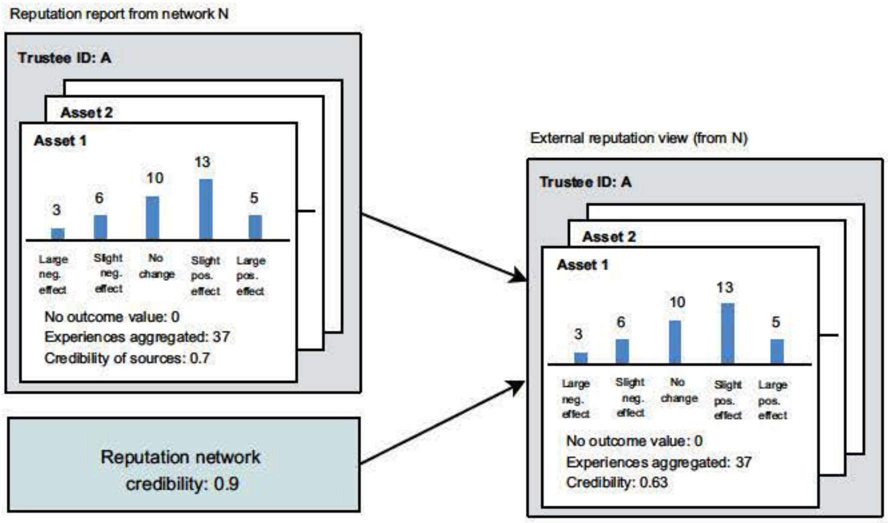

Figure 4.3: Combining agent-assigned credibility and network credibility by multiplication.

Credibility is a way of indicating that an information source is, for one reason or another, producing reputation information that is potentially not very useful as material for trust decisions. In order to build a feedback loop that benefits from earlier experiences with reputation sources, the source credibility values must be updated, for example through recalculating the trusted neighbourhoods around the actor [GOGH08] or discounting external information based on how it deviates from the local view [BB05]. We have not fixed the method to use for this for the purposes of this thesis, but allow different methods to be applied by different policies. The next section discusses a method for identifying and punishing false reports.

4.1.4 From subjective reputation to verifiable experiences

In the previous subsections, we have discussed reputation from the point of view of storing and using it in the local trust decision process. We now take a look at the feedback loop to the ecosystem: how locally gathered experience information is shared with other actors. The evolution of shared reputation information depends on new experiences being fed to the reputation networks and processed there. The sharing process differs depending on the reputation network, and therefore we discuss an example reputation system in this section as well.

A central problem in the process of experience sharing is the incentive and possibility of spreading false experiences, to defame competitors or to undeservedly promote others over them. Misinformation inhibits the reputation network actors' ability to assess other actors' behaviour, which limits the impact of reputation [Sol06]. The spreading of misinformation must therefore be detected and resolutely punished by a loss of reputation, and subsequently reduction of the actor's credibility as a reputation information source. This kind of a system of second-order punishment, i.e. punishing those who deal out unfair punishment themselves, is necessary in order for any community to scale up in size [FF03].

The capability to create this kind of punishment system depends on whether the experiences in question are treated as subjective statements, or objective and verifiable experiences.

In the usual case of reputation systems, shared reputation information is the *subjective evaluation* of the source, and unverifiable: there is no global monitoring that could see everything happening within the ecosystem, and the measurement criteria used by the source are private. In this situation, it can be questioned whether there is any point to even trying to evaluate the objective truthfulness of the information. Instead, we can focus on e.g. evaluating how useful the information was for the decisions made, after the fact. In other words, if a source constantly disagrees with up-to-date local experiences, it is less relevant for local decision-making. As a result, its source credibility value should be lowered locally.

In related work that considers source credibility, this type of evaluation is the norm. Besides testing whether external reputation information matches with local experiences [TPJL06], comparisons between multiple external sources [FKÖD04] can be used, with the assumption that the majority opinion is safe to follow. The most common approach in related work is to infer credibility solely based on other attributes of the source, and not even try to keep track of the actual spreading of misinformation. These indirect attributes can e.g. be the source's reputation as a service provider [KSGM03] or its social relations [SS02, GOGH08].

In addition to the loss of credibility, the source of useless information could also be locally shunned by storing experiences relating to the spreading of misinformation. They would fall naturally under negative impacts on the control asset; similarly, experiences on good information sources could be stored through this asset as well. Sharing this reputation information in the network would be problematic, however: it is a subjective judgement on the usefulness of subjective information, and its fairness is as difficult to evaluate as that of the original experience.

As we have stated earlier, spreading misinformation in reputation networks should be punished with loss of reputation in order to provide a form of social control in the ecosystem through reputation. On the other hand, sharing the judgement that this reputation information was useless for this trustor service simply opens up new problems: How to catch second-order misinformation, and how to avoid being wrongly accused for defamation if the majority happens to disagree with a given statement?

There are two solution options to choose between: settling for a subjective reputation system that does not provide social control, or paying the price for an effective reputation system. For the first option discussed above, we can accept that all experience information is relative and subjective, and simply pick the sources that agree with us — similarly to how e.g. movie recommendations work, where matters of taste are not challenged. This approach strips away the social control function of reputation, and shared experiences become simply a method of service advertisement. The result is a service recommendation system, which punishes only misbehaviour that is directed towards the majority, or particularly influential actors in the reputation network. In contrast, minority opinion only matters to those who have chosen to listen to said minority, and discriminatory behaviour against minorities goes largely unpunished. While we should clearly prepare to interoperate with these kinds of systems as well, the relativistic approach to reputation alone is insufficient for our purposes.

The second option is to define a form of experience information that has a truth value: it specifies a claim that is either correct or incorrect independent of the actor analyzing the claim. In other words, reputation information based on such experiences is *objective*.

In the case of electronic services collaborating under an electronic contract, objective experience sharing becomes feasible: if the contract was followed, experiences are positive, if it was violated, they are negative. More specific rules can and should be specified in the contract.

If an experience then claims that the contract was broken, verifying it becomes a question of verifying whether the breach actually took place. Due to the lack of global all-seeing monitoring or a globally trusted rule enforcer, which would solve the problem by itself, this verification process must be distributed. A secure marketplace enforcing good behaviour through monetary costs for misbehaviour has been proposed by e.g. Li and Martin [LM10]; it requires a globally trusted arbitrator who stores deposits for all actors and judges the dispute cases.

In order to make an experience verifiable, a nonrepudiable receipt must be provided of all business transactions, and shared experiences must be bound to a specific transaction so that these receipts can be tracked down. This way, peers other than the service provider and recipient can verify that an experience is bound to a transaction that has taken place, and that its outcome was agreed on by the transacting parties: in essence, they form an audit trail on the transaction. In related work, nonrepudiable cryptographical receipts have been proposed by e.g. Obreiter [Obr04], in TrustGuard [SXL05] and in NICE [LSB03].

The cryptographical setup of signing transaction receipts is reasonably straightforward, given that contracts require an infrastructure for digital signatures already. A major issue remains: the process of issuing receipts is not symmetric. In other words, the recipient of a service may refuse to provide a receipt afterwards, or claim that it never received the service. This is essentially a problem that cannot be solved by technology alone, and eventually, some form of trusted third parties — third-party witnesses taking the role of notaries, or eventually legal recourse to judge the likely outcome — must be involved for transactions where a denial of receipt would have particularly severe consequences.

If we assume that there are no trusted third parties to rely on during the collaboration, it is at best possible to limit the importance of the receipt that can be omitted: In TrustGuard, nonrepudiable receipts are issued already on the intention to use a service, rather than receiving it [SXL05]; that way, completion of the receipt exchange protocol will be necessary before the service is delivered. At this point the meaning of the receipt changes, and it can only be used to verify that an experience is bound to a transaction that has begun; it says nothing of its completion. For services whose use is continuous or can be meaningfully partitioned into phases, this idea can be generalized into a series of partial receipts. Partial receipts could be implemented through e.g. hash chains [Lam81], which have been proposed in the context of incremental payment for service usage [Hei08]. While the last partial receipt may still go unissued, the risk is limited to the service provision phase represented by that single receipt, rather than the entire transaction. Not issuing a receipt is a breach of contract, which can eventually be contested in the court, although this process is slow and costly.

In a separate article, we argue that that trusted third party “notaries” are both essential and worth their cost for the purpose of providing effective social control through reputation systems [RK11]. Unlike global enforcers, the notaries do not need to be trusted by actors beyond the two transaction partners, and suitable notary service providers can be found as a part of the population or negotiation process; a protocol for finding a jointly trusted third party has been proposed by Alcade [Alc10]. When an optimistic nonrepudiability protocol is used, notaries also only need to step in when there are problems with receipt issuance, as the partners can easily vouch for each other to provide verifiability when both follow the protocol [RK11].

Once verifiable experiences are available, the spreading of misinformation can be detected across the reputation network and punished by loss of reputation, not just a local loss of credibility. In the default case, any new experiences disseminated across the reputation network can be used as they are. Matters of dissemination in distributed systems are not central to this solution; to simplify the presentation, let us assume for now that a reputation network implementing this protocol contains the equivalent of a centralized storage, and that sources of experiences can be verified e.g. through signatures.

If a service provider wishes to rebut a negative experience concerning itself, it can use its nonrepudiable receipt as a proof. As a result, a new experience is issued by the disputer, referencing the transaction in question, and making the claim that the original source of the false experience had a negative impact on the reputation asset of the disputer. The evidence for this experience, in turn, is the combination of the false experience disseminated earlier and the receipt proving it false.

If a third party or the service recipient wishes to rebut a false positive experience, it will need the negative receipt. The service recipient, i.e. the issuer of the receipt, is the natural source for storing these, as the misbehaving service provider has no incentive to prove its own bad behaviour. Due to this, it should be the service recipient who reports the experience in the first place; in this case, disputing false positive experiences would not be needed. It is worth noting that with the bilateral receipt scheme, a collusion between a service provider and recipient allows them to generate unlimited positive experiences into the network which no one can prove wrong — when studied closely, this issue boils down to how we cannot verify, even by observing the message, monetary and eventually goods exchanges, that two collaborators actually did anything useful or just agreed to pretend that they did [RK11]. This is not an issue from the point of view of punishing unfair punishment: collusion is essentially its own form of collaboration, and neither collaborator has been wronged in the process. It does underline that local credibility analysis is still necessary: the two colluders can clearly opt to provide fake service to an honest trustor as well, at which point they should be punished with reputation loss.

Verifiable experiences make it possible to punish the spreading of misinformation in the reputation network, while due to factors such as the potential delay in the punishment process, they do not remove the need to analyze reputation information locally. Local credibility analysis of all incoming experiences remains the main technical recourse against misinformation: it is not necessary nor prudent to accept all experiences as equal, be they subjective or objective.

The Pilarcos architecture provides our trust management architecture with the necessary support to implement an objective reputation system that provides the desired social control in the service ecosystem. Additional information can be accepted from subjective reputation systems as well, through a local credibility analysis.

4.1.5 Tracking behavioural changes with reputation epochs

When analyzing experience and reputation information for decision-making, a central concern is whether the information is up-to-date, i.e. describes the current behaviour of the actor in question well enough to be useful in trying to predict its future behaviour. When faced with ten positive experiences and two negative, it can make quite a difference if the negative experiences are ancient and the actor's recent behaviour has been spotless, or if the good experiences are mostly older and the two negative are the most recent experiences available of the actor.

Whether it is possible to detect the order or timing of actions depends on the reputation information model. Maintaining the ordering information of experiences sets heavy requirements on the way experiences are stored and processed: treating each experience as a unique object with a timestamp, or with a position in a queue of experience objects, creates large data structures, which take an increasing time to process as the number of experiences grows. While this may not be an issue for offline data mining efforts, real-time decisions based on reputation information must ensure upper limits to the amount of data to process on the fly. In other words, either the experiences must be somehow compressed, which loses information, or old experiences must be purged after a while.

We have chosen to compress experience items into outcome counters. The compression tradeoff loses timing information. The basic reputation data structure places equal weight to all experiences, independent of the time they were gathered. While it would be possible to discount old information through aging factors or other often-proposed methods [JIB07, pp. 639], we find that most such methods in practice are either very costly or steadily lose information: they form a kind of fixed-size window to the past. The main problem with fixed windows, in turn, is that they cannot be easily adjusted at the time of the decision: data is already lost while it is gathered. For example, past transgressions can be completely erased from such systems by simply flooding the network with new experiences from low-value real transactions, or false experiences produced by colluding partners.

The main value of discounting old information is realized from reacting to changes in behaviour, i.e. by not allowing a good history to outweigh recent transgressions. Considered against this goal, we find that time is actually not the optimal measure for determining the weight or value of a unit of experience at all. Instead, the optimal measure is whether the experience brings new information; something we did not already know.

We can capture changes in behaviour by dividing reputation information into groups of abstract periods of a given type of behaviour. We call these groups *reputation epochs*; each epoch turns a new leaf in experience gathering. While the latest turn of behaviour is most interesting, it is also typical that there is very little experience on it; hence information from older epochs must also be included. The weight given to the current epoch determines the speed in which the system reacts to changes in behaviour. The number of reputation epochs also provides a measure of the consistency of the trustee: if experiences on the trustee are divided into a large and constantly increasing number of epochs, it indicates that the trustee's behaviour is not stable — or that it is not entirely fitting into any behavioural categories the system can detect.

While reputation epochs allow us to give less weight to old information similarly to time-based discounting, they are superior to the constant discounting approach in two aspects: First, as reputation epochs are based on behaviour changes rather than strict time periods, they fit the purpose of detecting when information is outdated in the sense of not being useful for predicting future behaviour. Second, the weighing policy between new and old information can be dynamically changed, and as no information is actually discarded, the oldest experiences remain available for later analysis: the reputation system can be configured to never forget anything without straining the decision-making process.

In order to not allow the epoch data structure to degrade into the aforementioned problematic model where experiences are stored as unique objects, we assume that the number of epochs will not grow without limit. For this purpose, we can set up epoch pruning processes that ensure that the number of distinctly stored epochs remains under control. For example, if it turns out that an actor's behaviour regularly fluctuates between two types of epochs, older epochs can be merged regularly, as the fluctuations then actually represent a different type of consistent behaviour in the long term.

In the general case, detecting changes between reputation epochs is similar to the reasonably well-studied problem of anomaly detection [Vil05a], and the algorithm can be based on for example a set of "normal" values learned from earlier data. In more specific cases, rather simple epoch change policies can be fitting. We present two example policies to achieve two different goals:

Load balancing: A service provider usually provides good service, but occasionally the service quality varies depending on the number of incoming requests. The first example policy should quickly react to a drop in the quality of service, as it also indicates a need for load balancing.

Oscillation detection: A service provider oscillates between good and malicious behaviour: first it collects good reputation, then it cuts corners in as many service transactions as it can. Whenever there is a fixed decision policy in use that is known or can be deduced by experimenting, the optimal attacker targeting the reputation system will collect just enough positive reputation to not be shut out of the community, which makes this kind of behaviour relevant to address. The second example policy should quickly react to this kind of change for the worse, but also take advantage of the service returning to normality.

In the load balancing example, we apply a simple dynamically learning algorithm: a window of n previous experiences is stored by the epoch change detector, and whenever a new experience falls outside the values present in the existing filled window, a new epoch is created. As normal service quality is indicated by the vast majority of experiences, the window is typically filled with such experiences. At the first drop in reputation, a new epoch and a new, empty learning window are created. While the disturbance goes on, the window is filling up with negative (or less positive) experiences. During this learning phase, when the epoch contains less than n experiences, new epochs are not created.

If the window (n) is set to be shorter than a typical disturbance, it will be full of negative experiences by the time the service returns to normal load, and a new epoch is started when the first positive experience arrives. This leads to a swift return to the service provider when it is no longer overloaded. For a more pessimistic, slow recovery, the window (n) can be chosen to be longer than a typical disturbance, which means that reputation is slowly regained within the newest epoch. Again, once the window fills up with normal experiences, the first sign of a negative experience causes a new epoch to be started. A limitation of this policy is that if the experiences indicating normal or overloaded states have some natural variation, new epochs may be created too easily.

In the oscillation detection example, the difference between good and malicious behaviour is simple to observe, as the experiences will be polarized: positive or negative. To allow greater variation in behaviour than the previous policy, we apply a static, specification-based epoch detection algorithm. We define two behaviour profiles: “good” and “evil”. The good profile covers positive experiences, the evil profile negative. Neutral experiences, or those representing unknown outcomes, fall in neither category.

Given these profiles, we define each ongoing epoch to be either good or evil, and the reputation epoch changes if an incoming experience matches the opposite profile rather than the current one. Neutral or unknown outcomes do not change the epoch, as they match neither. Again, the ongoing epoch can in principle be given full weight in decision-making. On the other hand, the attacker may respond by oscillating on every service request: cooperate, defect, cooperate, defect. To withstand this kind of behaviour, the number of epochs or the number of experiences in the current epoch should play a part in choosing a better weight division between the current and previous epochs, or indicate that the decision should really be delegated to a human user due to high uncertainty in the reputation information.

Even though the epoch detection profiles are statically defined like this, information about the “nature” of an epoch is not made available separately at the trust decision. Reputation epochs are primarily a method for ordering reputation information into units of consistent behaviour, and to divide the history of experiences on a trustee into partially outdated and more current information. While it may be tempting to see epochs as representing actual behaviour types, such as “reliable behaviour”, “regularly fluctuating behaviour” or “suspicious behaviour”, dynamically detecting meaningful behaviour groups is not feasible in the general case. For this purpose, only the divisions between epochs and the actual experiences stored in them are used in decisions.

The two above policies perform at their best when the central source of reputation information is either first-hand experience, or a single highly credible reputation network. On the other hand, sometimes experiences on an actor are only available through a low-credibility reputation network, where there may be errors in the experience information — either intentional misinformation or due to e.g. differences in measurement standards. To cover this scenario, we extend the oscillation detection case above with an additional requirement:

Conservative oscillation detection: A potentially oscillating service provider is only known through a reputation network where some experience reports are incorrect. The third example policy should be cautious in trusting reputation information that is out of the ordinary, and treat it as an outlier unless it is backed up by additional information.

Recall that in the earlier solution for oscillation detection, we specify in the policy that some observed behaviour belongs to the “good” category, some are categorized as “evil”, and some quality as neither; we follow this categorization in this conservative version as well. To address the additional requirement for a policy that is more resistant to noise and anomalous experiences, we apply the idea of sequential hypothesis testing [Wal45], which has been previously applied to limit the probability of overreacting in anomaly detection [JPBB04] as well.

The main idea of the solution is to not change epochs based on a single value out of the ordinary, because the experience flow is known to contain some erroneous reports mixed into the valid information. Instead, we should wait for additional supporting evidence before acting. The epoch is changed if the evidence amassed during a fixed waiting period mostly supports the change, or if overwhelming evidence for the change arrives even before the wait period is over. In this case, a change is warranted if the current epoch is “good” and the incoming experiences categorize as “evil”, or vice versa, and the amassed evidence translates into the comparison of the count of “good” experiences and “evil” experiences during the observation. We explain the resulting Algorithm 1 in more detail below.

Algorithm 1 Conservative oscillation detection.

```
 1:     ▷ Support and timer variables are initially set and reinitialized to 0.
 2: for each round do
 3:     if experience and epoch match then          ▷ (both are good / evil)
 4:         if timer == 0 then               ▷ No hypothesis testing is active.
 5:             skip to next round;
 6:         else
 7:             support–;
 8:         end if
 9:     end if
10:     if experience and epoch mismatch then ▷ (one is good, one is evil)
11:         support++;
12:     end if
13:     timer++;                            ▷ Advance the observation period.
14:     if support ≥ k then           ▷ This indicates overwhelming support.
15:         change_epoch();
16:         reset_variables();
17:     end if
18:     if support < 0 then
19:         reset_variables();
20:     end if
21:     if timer ≥ t then
22:         if support > 0 then ▷ Majority of the t votes supports change.
23:             change_epoch();
24:         end if
25:         reset_variables();
26:     end if
27: end for
```

In sequential hypothesis testing, a single experience out of the ordinary does not yet change the epoch. It only strengthens the hypothesis that the epoch should be changed, by a constant measure i; we set $i = 1$ for the purposes of this text (lines 9–11 in Algorithm 1). Similarly, an experience supporting the current epoch weakens the hypothesis by 1, if a hypothesis for changing the epoch has been established (lines 2–8). Again, neutral and unknown experiences cause no effect.

The amassed evidence is evaluated on lines 13–25 of Algorithm 1. For the epoch to change, either the change must amass support exceeding a given threshold k, or during a period of t consecutive experiences there must be more support for changing it than there has been for continuing the current epoch. In other words, if there is overwhelming support for the hypothesis, it is accepted and consequently the epoch is changed. If the evidence is somewhat contradictory, the algorithm enters an observation period and at its end, determines whether the hypothesis is true by checking what the majority of the amassed evidence supports. In the latter case, even the difference of a single observation in the other direction can flip the epoch, but we can observe that the support for continuing the current epoch is equally sketchy at that point.

Sequential hypothesis testing could be similarly combined to the window-based load balancing algorithm, to test the need to change epochs once the learning window has been filled. As a downside, this modification alone will not stop outliers in incoming reputation information from being stored as examples of normal behaviour during the learning process. Therefore, the algorithm would remain vulnerable to any noise in reputation flows.

The three example algorithms demonstrate the flexibility of the reputation epoch concept. We have performed illustrative simulations on them somewhat similarly to the experiments reported in Chapter 6; the results are reported in a separate paper [RHK11]. In the paper we demonstrate that the reputation epoch concept allows us to cover a considerable weakness in existing reputation systems by supporting timely reacting to changes in behaviour. Once the reputation epoch change policies are in place and the locally stored experiences are annotated with this consistency information, it is easier to specify the appropriate type of reaction based on what kind of decision is being made: for example for a short-term action, it is worth avoiding a service that appears to be malfunctioning right now, while decisions on longer-term collaborations can still take advantage of stored older experiences as well.

4.2 Adjusting to business situations through policy

TuBE combines the strengths of reputation-based and policy-based approaches to trust management. On one hand, it learns from the past through using up-to-date reputation information as a basis for risk and benefit estimation. On the other hand, it combines this information with local business rules and valuations, all easily adjustable to different and changing business situations through policy configurations.

A central aim of the design has been to separate the policies directing trust decisions that use the gathered information, and the policies directing the information gathering itself. Respectively, when we look at the effect of reputation on trust decisions, two different types of policies emerge: the trust decision policy and the reputation update policy. Both consist of a number of sub-policies.

The different policies in use are stored by a local policy manager, and queried by the TuBE guard, i.e. the trust decision subsystem, and the reputation management subsystem as they are needed. The policy manager exposes one interface for the user to configure the policies in use, and another for the policy-aware components of the trust management system to update their policies. These connections are depicted in Figure 4.4.

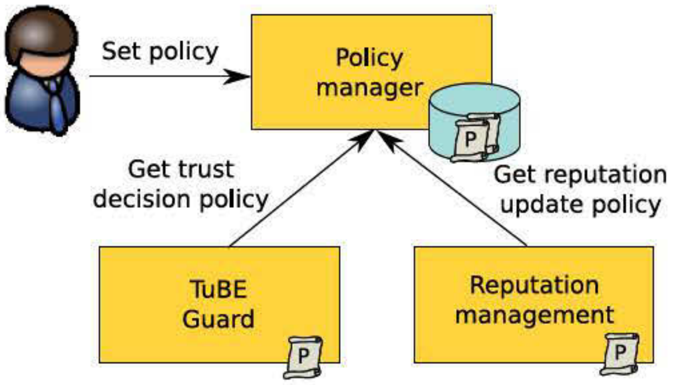

Figure 4.4: A policy manager stores the local policies in use.

4.2.1 Trust decision policy

In TuBE, the trust decision policy determines how the information stored in the system is transformed into a trust decision. Focusing on the point of view of reputation, it determines how reputation information stored in the system is translated into a risk evaluation, and through a comparison to the risk tolerance policy, converted into a trust decision on whether to allow the request or not.

Reputation information is stored in the TuBE reputation data storage, where it can be retrieved through a trustee identifier. Local and external experience views are stored separately; a *reputation view merging policy* determines how the two are combined. The merging calculates a new credibility to the merged reputation view by combining the local view's full credibility with the external view's credibility value. After the merge, *context filters* are applied to the resulting reputation view in order to adjust to special cases; for example, if the trustee is thought to be suffering from a reputation attack, its reputation may be temporarily increased with the help of a context filter until the reputation network in question recovers from the attack.

The merged reputation view is then transformed into a risk evaluation. In the default case, this is done directly by converting the experience counters to probabilities by dividing them with the total number of experiences, but an action-specific *risk template* can be applied when needed. As noted in Section 3.2, the probabilities are ratios of the given outcomeIt can for example reflect that an action is inherently low-risk and major negative outcomes are not possible, so their probability value may be transferred to the probability of minor negative outcomes instead. The risk evaluation is then also subjected to *context filters*, which can capture, for example, that abnormal activities in the business network have made certain resource-intensive actions temporarily more risky than usual. Now the risk estimate is complete.

In order to calculate the risk tolerance, action-specific importance values are determined by an *importance evaluation policy.* The value is subjected to *context filters*, again: for example, the importance or certain positive effect of any selling-related actions may be temporarily increased when low on storage space. The importance value is then passed as a parameter to the *tolerance evaluation policy*, which produces one or more tolerance constraint sets based on the importance value. *Context filters* are applied to these as well; monetary risk tolerance can for example be temporarily reduced due to running low on funds.

In the end, the risk estimate is compared to the risk tolerance constraints to determine which set of constraints it falls within; for example, two sets may indicate the ranges for "clear/routine deny" and "clear/routine accept", while any evaluations falling between them are deemed non-routine and forwarded to a human user to decide. Any messages relating to such deferred decisions are blocked by default until further notice. The outcome of either allowing or blocking the action is then forwarded to the Pilarcos monitor, which passes or blocks the message. This high-level process is depicted in Figure 4.5.

For most policies involved in the trust decision, a clear default approach is to not perform any changes to the given values unless specifically needed: all context filters, action-specific risk templates and importance valuations can be left undefined unless needed. There are a few natural default approaches for the reputation view merging policy; one example is a credibility-weighted average of the two views of local and external reputation.

The tolerance evaluation policy, and what kinds of constraint sets it produces, has a considerable effect on the outcome of the trust decision. We will compare different importance-independent tolerance evaluation policies

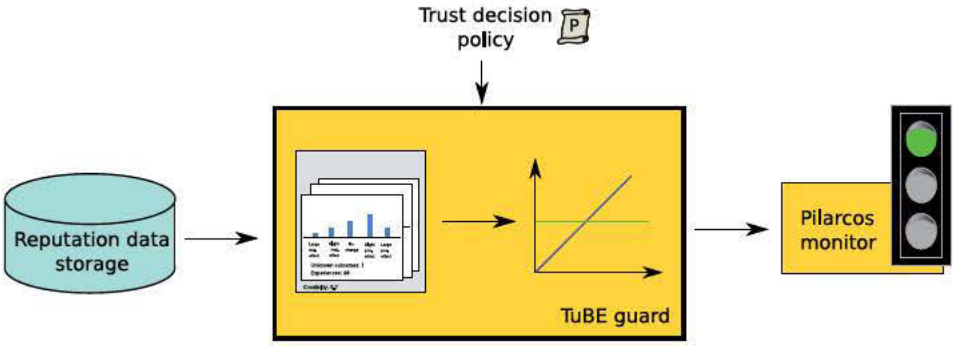

Figure 4.5: Transforming reputation information into a trust decision.

in Section 6.2. For this purpose, we have included two configurable types of tolerance evaluation policy in our simulation setup.

The *utility-based tolerance evaluator* policy adds up the existing experiences within each asset, using different multipliers for the different effects, and compares the value to a threshold. We liberally use the term "utility" here, as the policy can be interpreted as the trustor calculating an expected abstract cost or benefit, i.e. utility, to collaborating with the trustee, and comparing it to a minimum value required.

We divide the multipliers into the positive and negative multipliers k_+ and k_-, for positive and negative effects respectively, and major and minor multipliers, k_{maj} and k_{min}, for major and minor effects. The basic formula for the policy then becomes as shown in Algorithm 2, where p_x are the probabilities of specific effects for a given asset and a given trustee, and t_p is the threshold value.

Algorithm 2 Utility-based tolerance evaluator using probabilities (p).

```
**if** $k_{maj}k_+p_{maj,+} + k_{min}k_+p_{min,+} - (k_{min}k_-p_{min,-} + k_{maj}k_-p_{maj,-}) \geq t_p$
**then**
    accept(action);
**else**
    reject(action);
**end if**
```

As the risk evaluation also contains the total number of experiences with known outcomes, we can multiply all values by the total to transform each probability, i.e. ratio, back to the number of experiences with that outcome. This makes the behaviour of the risk tolerance formula slightly easier to understand, while also fitting into the more straightforward idea of the utility the trustee has so far provided to the trustor. If we multiply all terms with n, where n is the number of experiences with known outcomes stored in the risk evaluation, and replace the p_x in the formula with $u_x = p_x * n$, the above formula is transformed into one expressed directly through the number of experiences with a specific outcome, as shown in Algorithm 3.

Algorithm 3 Utility-based tolerance in terms of experience counters u.

```
if $k_{maj}k_+u_{maj,+} + k_{min}k_+u_{min,+} - (k_{min}k_-u_{min,-} + k_{maj}k_-u_{maj,-}) \geq t_u$
then
    accept(action);
else
    reject(action);
end if
```

The four different multipliers have a considerable effect on the outcome of the decision, and must be chosen with some care; we will demonstrate the behaviour of the policy with different multiplier combinations in the following experiments. The threshold value t_p or t_u sets the minimum value for when decisions are positive; if it is too high, potential partners must gain a high reputation before they are ever allowed into collaborations with the trustor; if it is too low, the trustor risks attracting malicious partners.

The same formula is repeated for all assets. The constraint set is fulfilled if and only if the outcome counters for all assets are above their thresholds. As analyzing the differences between the desired treatment of various assets goes beyond the scope of this thesis, we have not implemented different multipliers for different assets in this policy.

The *minor/major tolerance evaluator* policy builds on the same idea as the utility-based tolerance evaluator, but it will not allow conversion between minor and major outcomes, even through multipliers. Instead, it treats major and minor effect actions as fully separate. The policy has four multiplier values, and separate thresholds for minor and major outcomes.

The justification for not converting between different severity classes of experiences is that the actions involving outcomes with major effects are by nature different from those with only minor effects. If a partner turns out to be incompetent with critical actions but works well with actions demanding fewer resources, its reputation should be interpreted in a way that allows it to perform the minor actions but blocks it from major ones.

Using the notation from the previous formula, we get Algorithm 4. In it, t_{min} and t_{maj} are the threshold values for minor outcomes and major outcomes, respectively.

The two presented policy types are by no means a thorough selection

Algorithm 4 Minor/major tolerance evaluator.

```
if $k_+u_{min,+} - k_-u_{min,-} \geq t_{min}$ and $k_+u_{maj,+} - k_-u_{maj,-} \geq t_{maj}$ then
    accept(action);
else
    reject(action);
end if
```

of the possible decision policies. They do have a few central benefits from the point of view of this work: they are intuitively simple, their basic behaviour can be adjusted with configuration parameters, and the result can be easily illustrated by utility score plots. Also, a handful of specific policies that fall under either of these two types are sufficient to demonstrate the central considerations related to trust decisions: quickly reacting to bad experiences, giving more weight to high-value than low-value transactions, and overcoming information sparsity problems. We will return to this topic in Section 6.2.

4.2.2 Reputation update policy

The reputation update policy determines how external reputation information from different sources is incorporated into the external reputation view. External reputation information, recommendations, are delivered to the TuBE reputation management system from representative agents in reputation networks, and are translated into the local reputation format. The recommendations are then analyzed locally, and the reputation view is updated with the new information based on the reputation update policy.

The recommendations arriving can take two forms: they can be presented as single external experiences, or full reputation views, consisting of a set of experiences that should be evaluated and interpreted as a single whole. The presentation depends on what kinds of recommendation units the external reputation network communicates in: some translate easily to specific experiences, others into overall opinions. For example, a good credit rating value can be translated to n positive experiences, where n reflects the weight given to the information.

The simulation experiments in this thesis use a stream of external experiences from sources with different credibility. These streams of experiences are merged into a single external reputation view for the given trustee.

As our simulations focus on differences in credibility rather than in data format of different sources, we have opted to use only the experience format in the simulation. The handling of opinions and experiences is mostly similar, but with two things worth noting: First, due to the different nature of the sources, the policies handling opinions and experiences should be expected to be different. For this reason, opinions and experiences must be kept semantically separate in order to for example support policies of the form "always first merge these two views by averaging to represent a total of n experiences, then add any separate experiences into the result". Second, opinions are also associated to epochs similarly to single experiences, but with the difference that the reputation change from a previous recommendation must always be erased to incorporate an updated opinion from the same source. For example, the n positive experiences added due to a good credit rating must be removed when the credit rating turns bad, to ensure that the relative weight given to the single opinion source does not increase by n experiences at each update, unless the local policy setter deems it appropriate. The added complexity is a tradeoff in order to be able to handle a range of different information sources.

When a new external experience arrives from a given reputation network, it has been assigned a credibility value based on the analysis made by the representative agent in the reputation network. This credibility is first combined with the credibility of the reputation network itself by applying a *source and set credibility merger* policy, resulting in a single credibility value. The experience is then incorporated into the current external reputation view through a *reputation updater* policy, which is also applies a policy for updating the view credibility accordingly with an *experience to view credibility merger*, if necessary. The two policy applications have a one-way dependency: the reputation updater policy can choose to discard the experience altogether or modify it, in which case the same policy determines whether the view credibility should be updated at all. On the other hand, the more straightforward credibility merger policy remains unaware of the surrounding reputation update policy. The high-level process of external reputation updates is depicted in Figure 4.6.

A straightforward example of a source and set credibility merger policy, as mentioned earlier, is multiplying the credibility values, which are real numbers between 0 and 1. When merging the credibility values of the incorporated experience and the view, a similarly simple default policy would be to calculate a weighted average credibility between the view and the experience, using the number of experiences already stored in the view as its weight.

The central policy for incorporating new experiences is then the reputation update policy. We will compare different reputation update policies through simulations in Section 6.3. For this purpose, we have implemented

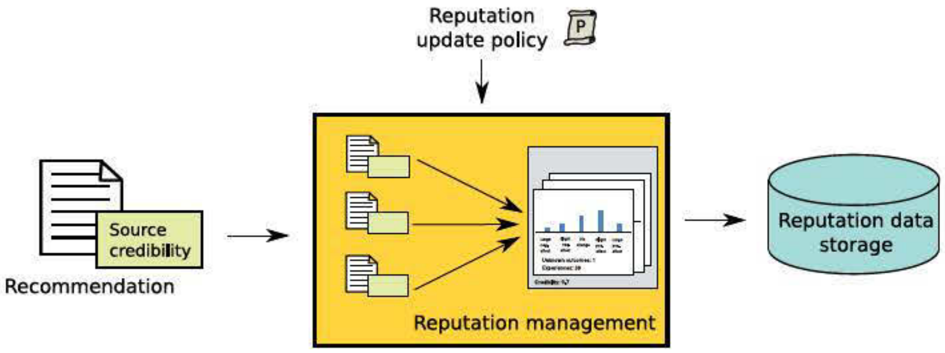

Figure 4.6: Credibility-aware reputation updating.

four classes of reputation update policy:

The *"accept all" reputation update* policy is a simplistic policy that incorporates all experiences, independent of their credibility. While this policy is the baseline for comparison, it can be used as such when the reputation networks connected to the system are all either trusted, or produce information of fixed quality so that credibility does not provide any basis for making update decisions. For example eBay [eBa11] stores all feedback provided, and has no model for credibility; all experiences received from such a system are equal in this sense.

The *weigh by credibility reputation update* policy incorporates experiences by weighing them with their credibility value: for example, two experiences of credibility 0.5 have the same effect as one experience of credibility 1. The approach gives more value to high-quality information, but is ready to accept more suspect input as well. This policy is special in that its implemention is by necessity stateful; we cannot incorporate the resulting real values into the reputation view directly, so the policy implementation must store "partial" experiences until they add up to full experiences. This approach effectively doubles the storage space used for external reputation, although the increase is negligible in a simulation environment with a low number of actors.

The *credibility cutoff reputation update* policy sets a fixed minimum credibility requirement for incoming experiences. It will not incorporate experiences if their credibility is below the cutoff level. The cutoff level is configurable. This policy only accepts high-quality information to ensure high-quality decisions.

The *match credibility reputation update* policy builds on the idea of a minimum credibility requirement for incoming experiences, but makes the cutoff level variable based on the quality of information already available.

Incoming experiences must have a credibility value that is at least as high as the view's overall credibility. For this policy, the credibility update policy for the view makes a considerable difference. For updating the view's credibility with the new experience, the weighted average update policy described earlier can be used. This policy aims to combine the best of both worlds: ensure that there is always some information available to make a decision, but once this has been achieved, it aims to protect the reputation system from being corrupted by suspect information.

In addition to the trust management policies, the business service is governed by a range of metapolicies which override any trust decisions. Decision-making is done on three levels: first, whether a collaboration is relevant in relation to the business strategy of the enterprise, second, whether it is valid, i.e. in accordance to for example the local privacy policy and legislation, and third, whether it is worth the risk [KRM08]. Trust decisions are positioned on the third level, and are only taken when the collaboration is already known to be both relevant and valid.

Policy configuration incurs an administrative cost when setting up the system. On the other hand, the policies are not changed on a daily basis, only at points where strategic changes are needed. We will return to the overall administrative costs of the trust management system in Section 6.1.5.

In summary, the trust decision and reputation update policies allow the trust management system to adjust to different business situations. The two sets of policies are kept separate in order to ensure that experience information can be collected only once, and then used in different decision contexts. We have specified some example policies to illustrate the possibilities of the trust information model; we will perform simulation experiments on these in Chapter 6.

4.3 Related work on reputation management

This section presents examples of reputation systems modelling reputation as a probability, which is prevalent in multi-agent marketplaces. It is close in philosophy to our approach of using experience information to calculate probabilities of different outcomes for subjective risk estimation. In the following section, we will discuss interoperability between reputation systems. We cover further related work under this topic as well.

As we have found in our survey work, the selection of proposed reputation systems is quite broad [RK05, RKK07]. While the reputation system of the eBay online auction site [eBa11] remains the best-known example system in use, reputation networks are extending beyond simple electronic marketplaces and private users. Reputation systems have been proposed to support partner selection also for virtual organizations [KHKR06] and Grid environments [vLAV05, TPJL06]; the requirements and available facilities of these kinds of environments are somewhat different from those in the marketplaces. Simulations similar to the ones we will discuss among the related work in Chapter 6 are used for evaluating reputation systems, particularly those directed towards multi-agent marketplaces.

Reputation systems proposed in the literature use a diverse set of methods for calculating reputation values, ranging from simple rating averages and different interpretations of probability to social network graphs. Jøsang, Ismail and Boyd provide further examples in their survey [JIB07]. The reputation values acquired can in turn be used as a threshold on whether to collaborate with a given trustee, as a method of ranking multiple actors to choose the most reputable partner, or presented as a decision aid to a human user [RKK07].

Probability as a concept can be interpreted either as a frequency of events, or more subjectively as a degree of rational belief or the plausibility of a statement. The latter definition is more abstract, and better suited for decision-making in contexts where probability values are not directly attached to random experimentation.

This division does not imply that frequencies or ratios are not used in probability calculations: reputation systems can and do calculate probabilities directly from the frequency a given outcome appears in previous experiences, as well as through more elaborate means. We can divide the process of translating experiences to reputation into three phases:

1. Pre-processing, in which raw experiences are manipulated based on additional information,

2. Transformation, in which the experience data is fitted into a probability value, probability density function or similar structure, and

3. Post-processing, in which the probability-based structure is further manipulated and possibly re-transformed into another value or format used in the actual decision-making or shown to the user.

For a simple example, let us assume an eBay-like marketplace where a seller has received 45 positive and 7 negative ratings according to a centralized reputation storage. In the pre-processing phase, the reputation engine may determine, based on separate evidence such as the reputation of the raters, that two of the negative ratings were uncalled for and therefore not usable. It therefore decides to include only 45 positive and 5 negative experiences in further analysis. In the transformation phase, the reputation engine calculates the ratio of positive experiences in order to transform the number into a probability value: $45/(45+5) = 0.9$. In the post-processing phase, the reputation engine changes this probability into a graphical representation, such as 4.5 stars out of five, to provide a more accessible decision aid to a human user.

The transformation phase, and often also the pre- and post-processing phases lose some of the information that has been fed into the system. In this case, 2 ratings were dropped from analysis in order to avoid skewing the result in the pre-processing phase. In the transformation phase, information about the total number of experiences was lost — the result would be the same even if there were three times as many ratings with the same ratios. Finally, in the post-processing phase, further details are lost in favour of simplifying the result to an understandable form to the user. Loss of information early in the process accumulates towards the end.

As an example of a more complex reputation calculation process, let us consider systems which model reputation as a probability distribution. As they typically pointedly apply the principles of Bayes' theorem [Bay63] in the transformation phase, these systems are also referred to as Bayesian reputation systems [JIB07].

The basic approach of these systems is to consider a sequence of transactions and their results as if it were a sample from a binomial or a multinomial distribution. Bayesian parameter estimation can then be used to discover the parameters of the target distribution [Nur06].

The Beta distribution is a probability distribution commonly used for these kinds of reputation systems; it is applied in the Beta reputation system [JI02] as well as many others, such as the model of Mui et al. [MMH02], Travos [TPJL06] and bHonest [Nur06]. A related binomial distribution is used by Perseus [Nur07]. The Beta distribution is a special case of the Dirichlet distribution, to which we will return later.

At the baseline, all information that is needed for discovering the parameters for the Beta reputation system is the number of transactions that ended well, and the number of transactions in total [JI02]. It is worth noting that as the transactions are not assumed to depend on each other on the probability distribution level, their ordering does not make any difference at the transformation phase. This means that discounting old information must be included as a part of input pre-processing phase, for example by constantly multiplying the aforementioned two transaction counters with a decay value whenever they are updated with new information [JI02]. Similar attention can be applied to include considerations for source credibility, relevance and other factors influencing the reputation update policy. For example Travos applies probabilistic methods for analyzing which reputation sources are likely to be inaccurate [TPJL06].

It is the differences in pre-processing that produce the core variations of different reputation systems based on the Beta distribution. However, the distribution itself sets a strong limitation to the format of reputation that can be used: as it is a distribution of binary events, all contract outcomes must be expressed in binary as well: either the trustee cooperated or defected. There is no way to express a partial cooperation or defection [RRRJ07a].

As noted by Jøsang and Haller [JH07] and Reece et al. [RRRJ07b], multinomial Dirichlet distributions are more flexible in this sense: they can be used to express cooperation and defection in multiple discrete outcomes of the contract. Jøsang et al. also propose using a Dirichlet distribution to produce a scaled outcome e.g. from 1 to 5, as an extension of the Beta reputation system [JH07]. Reece et al. use the Dirichlet distribution as a basis of observing a set of multiple, more independent contract outcomes, such as “goods were delivered on time” and “the quality of the goods was acceptable” [RRRJ07b]. Each independent outcome is binary: either goods were delivered on time or they were not.

Reece et al. note that a weakness with their application of Dirichlet distributions is in that all actors must observe an identical set of contract outcomes [RRRJ07a]; “not applicable” or “unknown” are not acceptable values in the binary outcome format. While this is not a problem with fixed scales, it is a severe limitation when dividing contract outcomes into categories: for example, while it could be relevant to measure whether installation support was available for high-tech goods, a similar measure would not be very interesting for plush toys. The group provides a solution to the limitation of fixed contractual outcomes: Kalman filters can be used to express the value “unknown” as well, which means that the reputation system can include observable variables that are not applicable to all transactions [RRRJ07a].

TuBE and the described reputation systems share the approach of calculating probabilities for positive and negative outcomes, but there are considerable differences in how the information is processed. In TuBE, we have chosen the simple approach of using ratios directly as base probabilities rather than fitting continuous probability distributions into the experience data available. The main reason for this is that the complexity of the model lies elsewhere; we store reputation itself as counters of experiences only, and calculate probabilities only at decision time, in connection to other policies adjusting the values.

Another difference is that TuBE specifically stores additional reputation metainformation in order to support the higher level decision on whether the trust decision is a routine case or not. As discussed in Section 4.1.5, we propose to divide the experience collection into periods, reputation epochs, as a way to limit information loss in the pre-processing phase on the specific issue of keeping track of behavioural changes over time. Epochs, together with credibility scoring and keeping track of the amount of experiences stored, provide important input in the decision phase on whether the automated trust management system should be determining that the decision is not really routine at all. If the information available is insufficient or otherwise suspect, the decision should be forwarded to a human user.

As noted by Jøsang et al. in their survey, a downside of using probability distributions is their complexity [JIB07]: users are unlikely to understand them, and yet they should trust the formalism to produce decisions similar enough to their own in order to delegate the decision process to an automated system. Further, if the pre-processing phase contains multiple different steps of discounting information fed into the distribution, and the post-processing phase re-flattens the distribution into a single scalar value, it becomes unclear whether the probability distribution formalism itself contributes much in terms of a sound theoretical basis either.

The transformation and post-processing phases are a part of the decision process rather than the reputation update process; activities in the pre-processing phase can belong to either group. When sharing reputation information between actors in a reputation network, and aggregating it into the TuBE system, it is preferable to use the raw experience information directly, if it is available. This will minimize the inherent information loss caused by the later processing phases.

4.4 Interoperability between reputation systems

One of the design choices in TuBE is to support gathering external reputation information from multiple different sources, which provides interoperability between reputation systems. Towards achieving this goal, this section will approach the issue through four subtasks:

- Identifying the different types of reputation information TuBE should handle,

- supporting pragmatic interoperability by assigning credibility to different types of information sources,
- supporting technical interoperability by converting reputation values from one format to another, and
- supporting semantic interoperability by addressing the fact that reputation values in different networks can mean different things.

We begin by studying the different **types of reputation information** that TuBE should handle. While related work on combining information from different reputation systems altogether has not emerged before this, there are some reputation systems which apply a combination of multiple types of information to produce the final reputation view. These systems can provide us with some insight on how to merge information from reputation systems with a focus on different types of information as well.

We find that the work of Huynh, Jennings and Shadbolt [HJS06b] on the FIRE reputation system acts as a representative example of a multi-sourced reputation system to discuss the attributes of different types of sources available. In FIRE, different information sources are combined into a single value based on a general measure of the reliability of the source type, and the overall quality of information available within that type. We will use the reputation types presented there as a basis of our discussion.

The FIRE model is designed for making trust decisions in open multi-agent systems. It is based on four different types of information: direct local experiences ("interaction trust"), external experience information gathered by the trustor ("witness reputation"), certificates of good behaviour presented by the trustee ("certified reputation"), and reputation based on actor class or role ("role-based trust") [HJS06b]. Hyunh et al. use a definition for trust which focuses on expectations of behaviour only; this falls within the definition of reputation used in this thesis.

Direct experiences are gathered from locally storing the results of the trustor's earlier interactions with the trustee, providing an equivalent of local reputation. External reputation, on the other hand, is gathered through the remaining three categories of information.

The first type of external reputation, witness reputation, is gathered by the trustor from agents who have interacted with the trustee. It is the most typical form of external reputation. As there is no centralized storage for reputation information in FIRE, the trustor discovers potential witnesses from the network using a referral system proposed by Yu and Singh [YS03]. The experience information is evaluated for overall quality based on e.g. the time of the transaction it originates from and source credibility. The resulting evaluation is then used to weigh the experiences against each other within this reputation type.

The second type of external reputation is based on the trustee actively presenting certified references to the trustor, such as digitally signed receipts of successful transactions [HJS06a]. The trustee-centric approach is familiar from certificate-based trust management and trust negotiations; it has been proposed earlier by for example Obreiter [Obr04], Srivatsa et al. in TrustGuard [SXL05] and Lee et al. in NICE [LSB03].

The third type of external reputation information is not based on experience, but on rules tied to the actor class the trustee belongs to. Membership in a specific class of actors or a given type of role can have an effect on the reputation of an actor; this information is particularly useful as a default value in the absence of sufficient experience information. Hyunh et al. note that the rules are domain-specific and set no limitations to how the role information can be acquired [HJS06b]. Services fulfilling particular roles may be expected to behave in specific ways according to the incentives of the role; for example, sellers may be expected to sell a product of slightly lower quality than agreed [HJS06b].

When we reflect the three types of reputation in FIRE to the TuBE architecture, the first-hand experiences are similar to local experiences, and the first two types of external reputation are simply two different ways of operating an external reputation system: one based on subjective statements, the other based on evidence. The latter sort may encode subjective statements that are stored by the target of the recommendation, or it can be verifiable and objective, as we have proposed in Section 4.1.4.

For considering the third type of role-based rules in TuBE, we find that the information could be applied as context filters for the reputation or risk value; they are not an actual reputation source in the sense of producing a flow of experiences from external actors. For example, the seller assumption would fit better as an adjustment of the risk of buying something rather than a source of nominally experience-based reputation information on the actor.

In addition to this, certain kinds of roles imply a certain level of *group-based reputation*, i.e. reputation inherited from other actors in the same group rather than a constant gain through a given role. For example, if the different services offered by a given large service provider are expected to behave similarly, new services by the same provider can be assigned a default reputation based on the reputation of the other services. This is based on internal reputation information, and due to the interdependency between multiple actors, it does not naturally fall under the jurisdiction of context filters. It could be implemented as a simulated “external” reputation information network based on internal experiences, but we find it more fruitful to separate a process of its own for handling the assignment of an initial reputation value to newcomers.

When a new service is found interesting, a three-level hierarchy of finding an initial reputation value to it is used: for new services from entirely new providers, a fixed default is applied — in the case of TuBE, storing no experiences is a good default. Group-based reputation can help assign an initial reputation to new services from known providers, essentially copying over the other service’s reputation in part or in full; rather than first-hand experience, this is a locally passed one-time recommendation. Therefore storing it as external reputation, with a credibility value below 1, is semantically preferable to storing it as local experiences. For services that are only unknown to the trustor, external reputation networks can be searched for additional information.

A related type of *membership-based reputation* can be defined as a recommendation an organization gives to all services (or providers) that have a membership within it. For example, an actor providing proof of its membership in a particular trader union can be given a default reputation value that represents how well the members of the union can be expected to behave — under the threat of being kicked out of the union for violating its rules, for example. Should such a union also actively vouch for its members, however, any insurance-like effects should be captured as a context filter for risk rather than as reputation.

Membership-based reputation can be interpreted quite generally: any third-party certification, be it for a government-certified seller, an evaluation for following process quality standards, or credit rating of a service provider, can be interpreted as a single specific but weighty recommendation from that third party. These certificates can be presented by the trustee in a form of trust negotiation protocol (refer to Section 2.3.1), or the information can be acquired directly from the source, such as a credit rating company.

Even the trustor organization can in some cases act as such an information source for itself, if it gathers and stores relevant information about partner companies that can be converted into a format usable by the trust management system. The trust criteria identified by Msanjila and Afsarmanesh, for example, combine multiple types of management-level information about a partner organization, such as its financial stability and technological expertise [MA07].

Certified, membership-based reputation, as discussed above, then becomes a special case of third-party certification that must be converted to the reputation format of TuBE. While the remaining types of certification do not fit very elegantly to the experience-based reputation model in TuBE, they can prove an invaluable source of base reputation when not enough local or external experiences are available. As such, they can be interpreted as equivalent to a given number of actual external experiences, and applied as a basis to which actual experiences from different networks are added to produce the updated external reputation view.

In summary, we have identified a number of different types of reputation in related work and existing structures in society, and ensured that they fit into the TuBE architecture and information model. First-hand and external experiences are the most straightforward category to deal with, while certificates, memberships in trusted groups and other aggregated opinions require a more elaborate conversion. The rule-based reputation effects of roles are best expressed through context filters rather than fixed reputation. Group-based reputation inherited from the social environment of a trustee is best dealt with through a separate reputation initialization process, which generally deals with assigning a default reputation to unknown actors.

Pragmatic interoperability considers the issue of choice: do we want to listen to a reputation source, and how carefully? In TuBE, this is reflected through reputation source credibility, which is assigned both to an entire reputation network and to individual sources within it.

Assigning credibility values to the different types of information sources presented above requires some insight into what kind of information can be acquired from them, and what kinds of incentives the reputation system architectures support for their actors.

For witness reputation, the decentralized approach of discovering recommenders allows the third parties to decide whether they wish to provide recommendations to the trustor about the trustee at a given time. A centralized reputation storage would, in contrast, allow only a single decision on whether to share the information with the network in general. All actors would also have to decide whether they trust the storage, and all provided information would be readily accessible.

With certified reputation, the problems are different: the trustee has full interest in presenting all positive information about itself, and acts as a centralized storage of information for itself. However, it has very little interest in providing negative information. This gives the experience information a strong positive bias. A recommender may wish to not provide a receipt for a good transaction to a potential later competitor, or retract its statement later; we have discussed the challenges of asymmetry and repudiability in Section 4.1.4.

The trustor may also have an interest to control who should know that it seeks information on a specific actor. In the case of centralized storage for witness reputation, and to a degree with role-based reputation, the trustor can acquire the information without anyone but the centralized reputation storage or the source of group membership information knowing about it. This may be particularly desirable when reputation is used in partner selection, and information is needed about actors who may otherwise not be aware of the kind of collaboration being set up. In contrast, the distributed witness reputation algorithm reveals the trustor's interest to everyone who is queried for a recommendation, and the certified reputation approach makes the trustee aware of the trustor's query directly.

The certainty and credibility assigned to each information source are tools to reflect the preferences of some types of sources over others when they are available. Reputation based on centralized certification, when combined with strongly authenticated sources, can be highly credible but semantically very limited in scope and slow to update, while the electronic equivalent of "marketplace gossip" can be readily available but prone to outside corruption. We have compared different credibility evaluation approaches in different reputation systems in earlier work [RKK07]. While various good mechanisms and methods could be identified from the studied systems, standard mechanisms and metrics for combining information from multiple reputation systems are missing. Standard-form metainformation of suitable granularity is also needed for evaluating the credibility of reputation information.

In summary, the architecture of a reputation system and the actors in the reputation network implementing it have a strong influence on the overall quality of information that can be received through the network. We can reflect these differences through credibility values, which supports pragmatic interoperability for reputation networks.

Once we have selected a way to represent an information source in TuBE, and assigned it a suitable credibility value, the remaining two issues concern the translation of the external reputation flow into the local data format.

To provide background on this goal of **technical interoperability**, we have studied simulation experiments in related work that aim to illustrate differences between reputation update algorithms that operate on different information models.

Before any reasonable comparison between models can be made, a conversion between them is needed. Besides illustrating behavioural differences of the reputation update models, it is a test of whether the target information model that the systems have been translated to can actually hold the central features of the different source information models. We have chosen the UniTEC experiments as an example illustrating this subproblem, and discuss the work in the TuBE context.

Kinateder et al. have designed the UniTEC trust and reputation model for electronic commerce [KR03, KP03]. The UniTEC reputation system is a fully decentralized peer-to-peer network which can store information about products or services in an arbitrary format; the model focuses on the distribution of the information and estimating source credibility. The system also employs privacy-enhancing structures, such as the use of multiple pseudonyms for each actor [KP03].

We have discussed UniTEC's approach to credibility estimation in comparison to other reputation systems in earlier work [RKK07]. We will not delve into details of the model further, but instead focus on how the generic elements of the model have been used as a basis of information model conversion between different systems.

Kinateder et al. have provided mappings between the UniTEC system and other models in a way that makes it possible to compare the behaviour of well-known algorithms built on somewhat different information models [KBR05]. They applied the approach to four trust models: the work of Abdul-Rahman and Hailes [ARH00] (with four discrete trust values and an uncertainty metric), Jøsang's Beta reputation system (based on Bayesian probability distributions) [JI02], the Regret reputation system by Sabater and Sierra [SS02] (using scalar metrics combined with reliability measures), and an algorithm originally designed for UniTEC [KR03]. After manually fitting the three other information models into the UniTEC model, Kinateder et al. simulated the behaviour of all four reputation update algorithms, including two variants of the Beta reputation system, with five different external experience streams. This final part of the experiment was quite similar in nature to the one we performed in Section 6.3.

The model conversion work could be compared to describing the reputation update policies of the different systems in a single policy language that is expressive enough. However, as we noted earlier, differences between information models may make the conversion impossible, and lossy at best, depending on the target model.

The relevance of this work to TuBE is twofold: On one hand, performing a similar conversion would allow us to compare the behaviour of various more complex reputation update algorithms expressed as TuBE reputation update policies. More importantly, a similar conversion would also demonstrate the expressive power of the TuBE information model.

Model conversion for reputation information is needed in TuBE in order for our system to interoperate with multiple reputation networks and collect information from them. For this goal, the reputation update algorithm used in the external network is or how it performs and behaves are both secondary. If the information produced by a given network is useful and relevant, we want to convert it to fit the TuBE information model and incorporate the experiences into the local reputation data storage.

The multi-asset reputation information model in TuBE is more expressive than the models we have considered in this work, and it is based on the concept of experiences from transactions. The complexity of most reputation models comes from translating this information to probability density functions, fuzzy values or trust categories for the *trust decision*; in other words, the central differences are seen in risk tolerance policies rather than the reputation information. The main challenge are opinion-based reputation systems, where the unit of recommendations is not an experience, but an aggregated, non-transparent value like “good credit rating”. Relating these to our discussion about reputation types, these are probably best interpreted as certification-based reputation, and primarily used as the basis of producing a reputation value for a newcomer at one time. Converting an aggregate value into experiences is a matter of defining how many agreeing experiences we would expect to see to create the same reputation effect.

In summary, the conversion of reputation information between two different information models supports technical interoperability between reputation systems. The experience-based, multi-asset TuBE model forms a good basis for converting experience-based data into, while for the transformation of complex, opaque values, manual mappings remain necessary.

As a last point, we must consider **semantic interoperability**: even if two actors use the same format for information, they may *interpret* it differently: what would be the objective meaning of an outcome scored “two stars out of five” as opposed to one scored “three stars out of five”?

In other words, two actors within the same reputation network may be met with the exact same situation, and while one considers it a positive experience because nothing bad happened, the other considers it neutral because the outcome was as expected. This is particularly valid for human users directly expressing their experiences: there are no standard scales for evaluations of feelings, and the users’ expectations have a strong influence on the outcome. Experiences may therefore not even be fully comparable for a single actor, due to their expectations and valuations fluctuating over time.

The inconsistency caused by changing expectations can be partially alleviated in the case of automated monitoring, as monitor rules define experiences explicitly. Local policy on how to interpret monitor events can still vary over time and particularly between service providers, however.

Systematic bias, such the eBay users' bias towards positive ratings[1], should also be taken into account when converting information from one network to another. One form of systematic bias can be caused by reciprocation between users [MMH02], which some systems support more strongly than others. For example, reputation networks where users cannot see who has rated them do not directly encourage the users to return the positive (or negative) rating in kind.

The differences in interpretation between single users are particularly challenging to account for, and yet overcoming the semantic gap is integral for shared experiences to be at all meaningful. Reputation systems such as eBay provide a means for users to attach a short explanatory message to each rating [eBa11], which helps other users willing to dig through the rating data to understand the numeric rating. When the data is used for automated decisions or recommendations, however, free-form text is not particularly useful; the semantics of the ratings themselves become much more important.

Dondio et al. have proposed a model for translating ratings between two rating models in recommender systems [DLB08]. Besides looking into differences between the models themselves, they also discuss detecting user-level differences in using numeric rating scales. Users can exchange information about what kind of attributes they focus on when rating actors in general by exchanging ratings for specific stereotypical scenarios. This stereotype information is then used to automatically adjust the ratings to better fit the interpretation of the user receiving the recommendations.

Gal-Oz et al. propose an architecture and model for sharing reputation knowledge across virtual communities [GOGGF10]. A matching component of the architecture is in charge of a task similar to that described by Dondio et al., although the approach does not focus so much on user-specific information. The component computes how well the communities match each other based on community-wide information, converts reputation values and scales them based on statistic information on rating differences, and

[1]It appears that eBay users give positive ratings for all transactions that ended at least as expected, and neutral ratings if there are only minor complaints, such as the delivery being late or the product not entirely matching its description. As a result, 99.1% of the feedback provided by buyers was positive [RZ02].

maps attributes of reputation between communities based on a mapping of each community's attribute set to a generic attribute set. The authors also discuss different alternative protocols for data exchange between communities, and their implications for information accuracy and user privacy.

The technical conversion between different reputation system data formats and the TuBE reputation format is an item of future work. Semantic conversions specific to actors in a reputation network are best done within the external reputation systems themselves, although a systematic bias in an entire reputation network can be taken into account during the conversion of the network's output to TuBE.

To summarize, achieving semantic interoperability requires methods to overcome differences in measurement scales on the interpretation level, including systematic bias. Users of the same reputation system may interpret the experience values in use differently as well, which creates a more local bias. Automated experience reports based on shared rules, such as the contract-based scheme proposed in Section 4.1.4, are more resistant to bias, but we cannot expect all reputation sources to converge to using the objective model.

In conclusion, we have discussed different types of reputation information that should be processed by the TuBE system, and reflected these to the three aspects of pragmatic, technical, and semantic interoperability to achieve between reputation systems. We find that the TuBE information model forms a good basis for plugging in the basic reputation information from different kinds of systems, while many of the more complex models for decision-making can represented through risk tolerance policies. Semantic interoperability must be considered for each reputation network separately, although it is promising that some more general approaches are emerging in related work.

4.5 Chapter summary

Two central requirements set for the trust management system are that the system should learn from first-hand and globally shared experience, and the trust decisions should be easily adjustable to different and changing business situations. The experience-gathering process has been separated from the trust decision process, as they are directed by different choices of policy.

In this chapter, we have presented the aggregation process for reputation: where reputation information is collected from, how it is managed and how it affects trust decisions in practice. Local reputation is built from direct observations through the trustor's monitoring facilities, where monitoring rules specify the trustor's interpretation of noteworthy events to experiences. External experiences, on the other hand, are incorporated from multiple sources, which must be both translated into the local format and critically evaluated for credibility. The separation between local and external experiences is maintained until decision time, which makes it possible to adjust their relative impact on decisions at any point.

To fulfil the second requirement of a trust management system, we have presented the central policies that allow the system to be adjusted to different business situations. The business situation and valuations direct all decision-making within an enterprise, and this must be supported by any automated decision system for it to be trusted to handle even routine trust decisions without supervision. Sensible default policies are provided for any adjustment points that are not needed by a particular enterprise in its current situation, but it is important that they too can be changed if needed later on.

Chapter 5

The TuBE system architecture

In order to evaluate the functionality and operation costs of the architecture and algorithms proposed in earlier chapters, we have implemented them as the TuBE trust management system, and present the result in this chapter. We will first describe the core system components, and show how they are used in our simulation experiments, which will be discussed in the next chapter. We then extend the discussion to the overall TuBE system architecture, with connections to external systems such as the Pilarcos monitor, reputation networks and configuration tools. In the third section, we discuss how the behaviour of the system can be adjusted according to the needs of the user. The next chapter focuses on evaluation.

5.1 Core system and simulation setup

The core functionality of the TuBE trust management system has been implemented in Java. The 5000-line implementation focuses on supporting behavioural simulations rather than providing a fully distributed simulation environment; it consists of plain Java classes communicating directly with function calls.

In accordance to the design goals described in Section 2.2.4, the data structures and algorithms used are reasonably simple in order to remain understandable to the user. As the complexity of the system is limited to the policy objects used to configure it, the configuring user is able to balance between simplicity and expressive power as is appropriate for their application.

The implementation has four central system classes in charge of processing information and applying policies, three passive data storage classes and a number of limited-scope policy classes, which are shown in Figure 5.1.

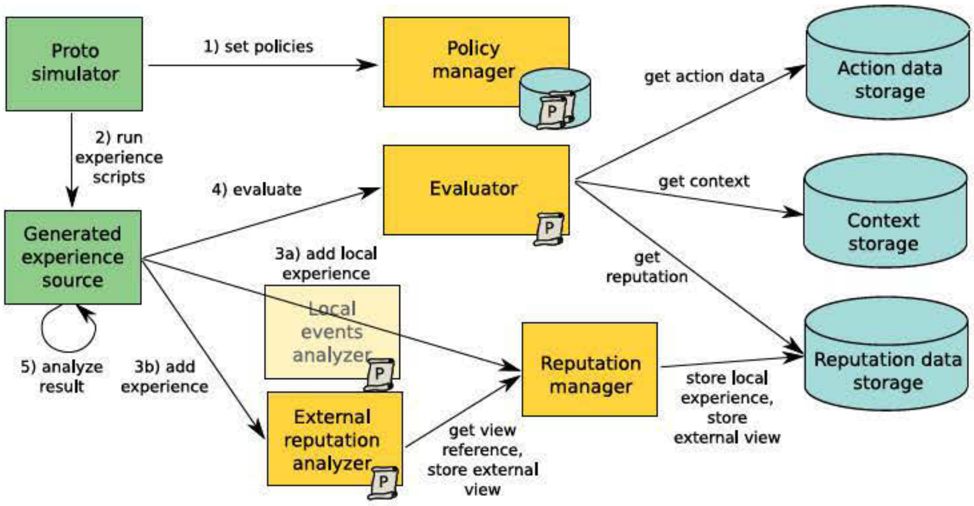

Figure 5.1: The core system and connections to the simulator.

The services provided and repository data contents are elaborated below. The simulation component involves an additional pair of active classes, shown in the left side of the figure. The simulator classes connect to the system classes and imitate connections made by external systems, including producing experience items based on hard-coded scripts and prompting for decisions after information updates. Additional support classes, representing e.g. data items, are omitted from the architecture image; they are listed in Table 5.1 on page 109. The policy classes are elaborated on in Table 5.2 on page 110.

The four main system classes are the Policy manager, Evaluator, Reputation manager and External reputation analyzer. The *Policy manager* is in charge of storing and distributing policies, which are denoted by "P" icons in the figure. Policy distribution has been implemented as each policy-driven class storing a reference to the Policy manager, and re-requesting the current policy each time they need it. The policy classes are direct Java class implementations of the policies they represent.

The *Evaluator* processes reputation, risk, importance and tolerance information for a trust decision according to the basic algorithm presented in Section 3.2. In the process it applies four policies:

- a reputation merging policy defines how to merge local (U^{local}) and external (U^{ext}) reputation into a single unified view (U^{merged});
- a risk evaluation policy defines how information in the action data storage is applied to form the base risk (β function) for an action,

which is then combined with the reputation view (U^{merged}) of the actor, providing probabilities ($\mathbf{p}_a$) for different outcomes in the risk evaluation (R);

- an importance evaluation policy similarly defines how information in the action data storage is applied to form the importance (I) for an action; and
- a tolerance evaluation policy ($\Phi_{\mathrm{T}}(\mathrm{I})$) transforms the importance of the action into risk tolerance formulae (T).

The result of the produced evaluation is a combination of risk estimate and a set of risk tolerance constraint sets; when the risk is passed as a parameter to the tolerance constraints, the final trust decision can be made based in what constraints the risk fits within, such as “definite no”, “definite yes” or “gray area”, which denotes a need to delegate the non-routine decision upwards.

In order to produce a trust decision, the Evaluator retrieves information from all three data storage classes: It retrieves base risk formulae from the *Action data storage*, which stores a grouping for each action and connects each group with information needed to build specific base risk formulae for them. For our simulation, these are not set nor used, as the risk evaluation policy in effect during the simulations considers all actions as equal. The action objects carry reference to the storage holding their data. From the *Context storage*, the Evaluator retrieves active context filters to apply to the different decision factors. While these filters are also policies in essence, their number and internal priorities vary, which makes it more natural to interpret them as input to the decision rather than a part of the policy configuration. Therefore they are also not in the domain of the Policy manager, which upkeeps one policy per policy implementation point. The available parameters for determining whether to apply a context filter or not are the trustor, trustee, action, contract id, and time.

From the third data storage class, *Reputation data storage*, the Evaluator retrieves reputation information, which is tied to the trustee. The Reputation data storage forms the connection between the decision-making in the Evaluator and reputation management done by the External reputation analyzer and Reputation manager.

The *Reputation manager* is the key interface for updating the Reputation data storage. For the purposes of our simulation, the Reputation manager simply passes onwards information for storing; it does not feed back local experiences to the external reputation systems, or compare local and external experiences to analyze the credibility of external information sources, as this part of its tasks is not relevant for the simulation setup. Hence it does not use any policies in this implementation, either; we discuss the full interactions of the reputation management classes further in the next section.

The *External reputation analyzer* receives reputation information, translated into the TuBE internal format, from external reputation sources. It applies three policies; the first one to to determine how to merge the credibility attributed to the incoming information by the recommender and the credibility of the source network, the second to determine what experiences to merge into the current reputation view and how, and the third to set the combined credibility for the updated reputation view.

The two main simulator classes are the Proto simulator and the Generated experience source. The *Proto simulator* runs the high-level experiment, setting decision and reputation update policies in the policy manager and identifying which predefined simulation scripts to run with the policies. The *Generated experience source* runs the assigned experience script and alternates between invoking the Reputation manager or the External reputation analyzer to store a single experience, and then calling the Evaluator to produce an evaluation for a trust decision.

As a final step, the Generated experience source analyzes the evaluation result and produces a log file entry for processing the simulation results. For the simulations described in this thesis, the stored result depicts the current reputation of the actor during the given round and a “score” calculated with the help of the risk tolerance formula; the classes implementing the risk tolerance formulae each provide their own support data for this purpose. For example, for a tolerance policy requiring that there are more positive than negative experiences, the Generated experience source logs the value (positive experiences − negative experiences). When this value is positive, the trust decision is positive as well. This information has been used to produce the graphs in Section 6.

Besides the system-level classes, there is a set of support classes. Beyond the policy classes, the support classes have limited functionality, and mainly serve to represent different data structures. A summary of the support classes is presented in Tables 5.1 (the `tube` top-level package) and 5.2 (the `tube.policy` package) on the following pages. The system classes reside in the package `tube.system`, with external reputation management grouped under `tube.system.externals`. The simulator classes reside in `tube.system.clients`.

Overall, the prototype is designed to have a high level of internal indirection: for example, classes that only need to be able to match Action

Table 5.1: The `tube` package: basic data structures and indirection classes.

`tube` - basic data structures and indirection classes	
Action	Representation of an action.
ActionId	A unique id for an action.
ActionParameters	The parameter data for an action.
Actor	Representation of an actor, such as a trustee.
ActorId	A unique id for an actor.
Asset	Representation of an asset and static data on the default assets.
Compressed-Experience	The experience counters of a reputation view.
Context	The context of a given action; a set of ContextItems.
CounterId	A class for creating locally unique counter-based identifiers.
Credibility	Representation of the credibility of an information source or reputation view.
Evaluation	The result of a trust decision evaluation, carrying the risk and tolerance data.
Experience	Representation of a single experience: outcomes per asset.
ExperienceId	A unique id for an experience, for referring to open experiences.
ExperienceSet	A data structure for storing and upkeeping open experiences.
External-ReputationView	Representation of an external reputation view.
Importance	Representation of an importance evaluation for the action.
LocalReputation-View	Representation of a local reputation view.
Outcome	Representation of an outcome, and a static list of the different possible outcomes.
Reputation	Result of a reputation evaluation, carries the merge of local and external reputation views.
ReputationView	Representation of a reputation view; functionality common to local and external reputation views.
Risk	Result of a risk evaluation, carrying the probabilities of different outcomes for different assets, credibility of the merged reputation view and representation for the amount of information stored in it.
Tolerance	Result of a tolerance evaluation, carrying a set of Tolerance-Constraints.

Table 5.2: The `tube.policy` package; policies in the system.

`tube.policy` - abstract classes for different policies in the system	
ContextItem	Implements a context filter rule (a modification routine and conditions for executing it) for a decision factor or several.
ExpToView-CredibilityMerge	Implements a rule for recalculating an external reputation view's credibility value when a new experience of a given source credibility is merged to it.
Importance-Evaluator	Implements a rule for calculating the importance of a given action.
Policy	Inheritable for all policies; identifier and priority handling.
ReputationMerge	Implements a rule for merging local and external reputation views.
ReputationUpdate	Implements a rule for merging a new experience to an external reputation view.
RiskEvaluator	Implements a rule for calculating the risk of a given action, given the trustee and its reputation.
SourceAndSet-CredibilityMerge	Implements a rule for merging the credibility of a source reputation network overall and the credibility value set by the network itself.
ToleranceConstraint	Implements the rule for accepting a risk to be within a given risk tolerance constraint class.
ToleranceEvaluator	Implements a rule for calculating the risk tolerance of a given action, given its importance.
`tube.policy.defaults` (actual policies) and `tube.policy.dummy` (testing tools) extend the abstract classes to implement a set of policies for experimentation. The policies used in the simulations are described in Section 4.2.	

identifiers to each other assume little about what form the action's parameters take, or even whether the identifier itself is an integer or a string of characters. This kind of encapsulation results in a high number of support classes with relatively limited functionality, while in return hiding the internal structure of data items from classes which do not need that information.

The simulation environment has been implemented as a standalone tool. The simulation setup is simple, involving only single-threaded processing.

For distributed and more large-scale simulations, a set of jointly used data structures currently implemented as unsynchronized Java HashMaps will need to be changed into interfaces to thread-safe data storage with capacity for efficient searching also for large amounts of stored data.

TuBE in itself receives its input and configuration through connecting to multiple external systems, such as the Pilarcos monitor, but for the simulations these connections have been replaced by the simulator classes. These classes produce a scripted stream of experience and prompt for trust decisions with given policy configurations after every experience item stored. The next section moves beyond the simplified simulation setup to discuss the full TuBE architecture as a part of the Pilarcos middleware.

5.2 The system architecture and interfacing

The core TuBE trust management system interfaces to multiple systems for connecting to reputation flows, receiving triggers for trust decisions, and configuring the system, including policy adjustments. The implemented standalone system connects only to simulator classes replacing these systems, but the additional connections set requirements for TuBE to interoperate with other systems, which are to a degree external to the core trust management functionality.

TuBE connects both to the surrounding Pilarcos middleware, which is considered an "external" system in the sense that TuBE has very limited awareness of its internals, and to systems which are not utilized by other parts of the middleware. These connections are summarized in Figure 5.2.

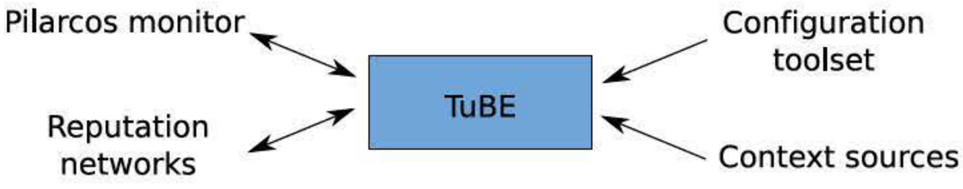

Figure 5.2: Connections between TuBE and external systems.

Within the Pilarcos middleware, the trust management system is most tightly integrated with the *Pilarcos monitor*, which both uses TuBE as one of its decision-making units, and also provides TuBE with message data which is transformed into local experiences.

A second connection to the middleware whole is created by TuBE's configuration interface. Through this interface, a *configuration toolset* can be used to make automated and manual configuration changes to the decision and reputation update policies, to manage action data such as base risk information, and adjust rules governing context filters. The connection is one-way in that TuBE itself remains unaware of the internal operation of the system providing the configuration. In the simulation system, all configuration options are set directly by the simulation classes.

TuBE also has two connections to external systems that are independent of the rest of the Pilarcos middleware. The first one is the system's representation in *reputation networks*, which provide external reputation information and to which local experiences are communicated. TuBE needs a specialized representative agent in each reputation network it connects to, as each network may use a different reputation system, and consists of its own set of actors causing different policy needs for participation. The agent operates as a member of the reputation network on TuBE's behalf, and communicates with the core TuBE system directly.

The second independent connection is to *context information sources.* The source for manual context adjustments can be integrated into the configuration toolset, but in more complex environments, context filters are generated and re-prioritized automatically based on triggers from e.g. an Enterprise Resource Planning (ERP) system. In the latter case it is natural to separate the configuration tool for fixed policies from the more dynamic and partially automated context management. In the simulation setup, context is left unconfigured and remains fixed. Figure 5.3 presents the TuBE system architecture in full.

The first and foremost connection between the TuBE decision making system and the Pilarcos middleware arises through the *Pilarcos monitor* [KMR05]. The monitor has been implemented to be plugin-based in order to make it straightforward to add new monitoring functionality to it. The monitor plugins can either proactively affect the messages passing through the monitor, or passively process them for information. This facilitates connecting TuBE into the monitor in the form of two plugins: a Trust decision plugin, and an Experience gathering plugin.

The *Trust decision plugin* is a proactive monitor plugin, which means that the monitor forwards message traffic it intercepts to the plugin, and waits for the plugin to make a decision on whether the message is allowed to pass or not. The trust decision plugin then consults its decision trigger rules to determine whether the message warrants a new trust decision, or if it is part of a message sequence for an action that has already been decided on. An example trigger could be the arrival of a specific type of message, such as a product order; the rules are kept very simple for efficiency reasons. If necessary, it makes a request to the Evaluator to

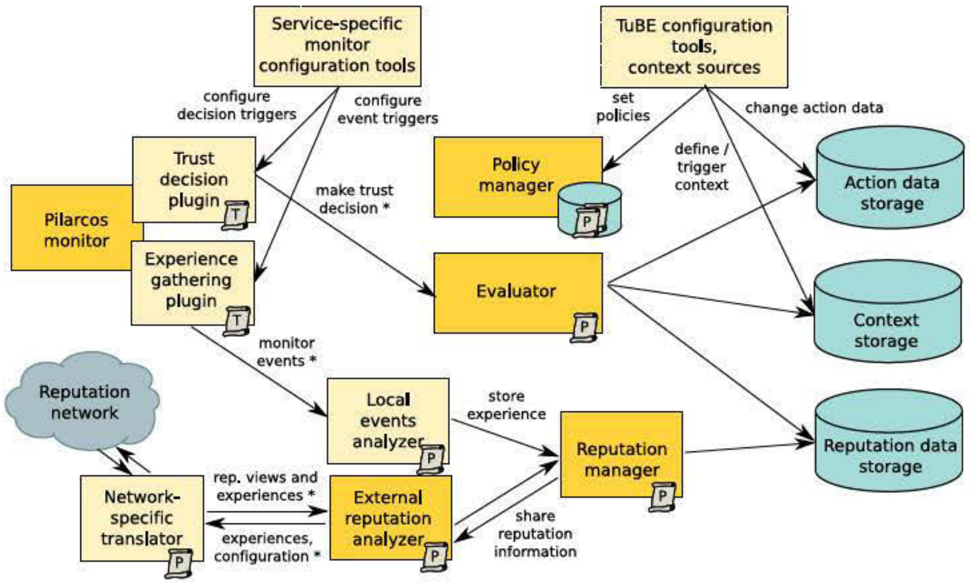

Figure 5.3: The TuBE architecture in full. The light yellow modules are replaced by simulator classes in the simulation setup. An asterisk (*) denotes a many-to-one relationship.

produce a trust decision, similarly to how the scripted simulator requests one in our experiments. The result of the decision determines whether the plugin approves the message for passing along or not.

The *Experience gathering plugin* is the initial source of local reputation information. It is a passive monitor plugin, which means that the monitor forwards message traffic to it, but does not wait for the plugin to respond to it in any way. The plugin holds rules on what kinds of messages to monitor for, and what information to gather from them. When such messages arrive in the plugin, they are translated into simpler monitor events containing the necessary information for experience gathering, and forwarded to the Local events analyzer.

Both the monitor and its plugins are replicated for each business service instance, while a single instance of the TuBE core system can be used to provide for multiple services within the same organization. This means that calls to make trust decisions and monitor events describing the business service activity can arrive to TuBE from multiple sources. For load balancing purposes, it is also possible to split the decision-making system (the upper half of the architecture figure) from the reputation information analysis (the lower half of the figure) and replicate them separately. They can for example serve different groups of services, as long as the reputation data storage is synchronized between them.

The *Local events analyzer* accepts event information from Experience gathering plugins, prioritizes the different events in relation to each other if needed, and translates them into outcomes of the action they are connected to. While an action is ongoing and has already had some effects that may still change based on later messages, the analyzer stores local experiences as open experiences via the Reputation manager. Open experiences can already be used for trust decisions, but unlike completed, closed experiences, they can be updated if later monitor events demonstrate a need for it.

For example, for a buying action, the trustee may fail to pay on time, which may in itself have a minor negative effect. If the payment fails to arrive altogether, the effect is major, but if the partner compensates for the delay, the effect may yet become positive. In some cases, the time during which compensation is possible is so long that it pays to store how the action seems to have ended while the outcome can still change. When further updates are no longer accepted, the Local events analyzer notifies the Reputation manager that the experience can be closed. Open experiences must be stored as independent objects, which will eventually slow down the decision-making process if their number grows.

The configuration toolset we referred to earlier in this section is divided into two parts at the top of Figure 5.3. *TuBE configuration tools* set policies used by the core TuBE system, and keep the Action data storage up to date. They are also used to define context filters, while the filters can either be triggered manually through the tools, or automatically through *Context sources*, which can be for example other parts of the Pilarcos collaboration management middleware. This strategic configuration adjustment can be used when the trust decisions need to be adjusted on the changing phases of the collaborations managed by the middleware; these phases were discussed in Section 2.2.3.

All policies within the core decision system are managed through the Policy manager; classes using them are marked with scrolls tagged with "P" in the figure. In the simulation setup, the Proto simulator class replaces the policy configuration interface. It passes policies on directly as their Java implementations, while a full policy configuration setup will need to transform the policies from policy language to implementation. As the policy languages are only ever seen by the configuration toolset and transformed to Java for distribution and use, applying a different language only requires a new interpreter in the configuration tools. Due to this flexibility, defining a specific set of policy languages is outside the scope of this thesis.

The *Service-specific monitor configuration tools* are used to upkeep the trigger rules for requesting trust decisions and generating monitor events in the TuBE monitor plugins. The tools can either be a part of the generic TuBE configuration tools or operate entirely separately. To differentiate these rules from the policies used by the core TuBE system, they have been marked with scrolls tagged with "T" in Figure 5.3. With suitable annotations in place, the monitor configuration can to a large degree be extracted from the business network models discussed in Chapter 2.1. Despite this connection to high-level configuration, monitoring rules are very simple and straightforward to ensure their efficient implementation.

The fourth and final connection, to external Reputation networks, is depicted at the bottom of the figure. Each *Reputation network* may use its own reputation system and data format, and different policies can be set in the External reputation analyzer for how local information is shared and new information accepted from these external networks.

For each Reputation network, a *Network-specific translator* stands between the network and the TuBE external reputation management. The translator converts information from the native format of the network to the TuBE internal format, either as full reputation views, or separate experiences. The translator must also be notified by the *External reputation analyzer* of what actors' reputation it should observe, and how their identities in the network are converted to actor identities in TuBE. This also forms the extent of the policy configuration of the translator; the final decision of what experiences are considered credible enough to include is left to the External reputation analyzer. A centralized decision is needed here, as only the analyzer has information on all the networks, including their credibility, as well as information on the gaps in existing information that must be filled in at different levels of urgency. The External reputation analyzer receives knowledge of what actors experience is needed on from the Reputation manager, which also has an overview of what kind of local information is available.

Besides extracting information from the Reputation network, the External reputation analyzer also forwards local reputation information to the network through the translators. The *Reputation manager* observes its own policy on when local experiences have changed enough to warrant an update; this information is forwarded to the translators via the External reputation analyzer, which applies its own policy on whether specific information should be shared in a given network. Only local experiences are shared: once external reputation information has been accepted into the TuBE system, it should not be sent out again, as this can create problems with giving some opinions additional weight by repeating them from one network to another [JMP06]. In the end, receiving the echoes of such repetitions from the reputation network corrupts the trustor's own reputation information.

Another task for the Reputation manager is to determine when a change in incoming reputation information is credible and strong enough to warrant an epoch change. Local and external reputation views follow the same epochs, as they are assumed to depict the same actor's behavioural change. While the actor may only have changed its behaviour in a single network and not be planning to change its behaviour towards the observing trustor, the social pressure function of reputation supports reacting to the change nevertheless. In addition, many reasons for a behavioural change, such as too much load, lack of funds or a service having been hacked, can be expected to reflect on the local trustor as well. For external information, credibility and the amount of information supporting a change are crucial in determining whether a change in incoming experiences actually depicts a change in behaviour or whether it should simply be interpreted as an outlier or an error, which should not cause a reputation epoch change.

In the simulation setup, the Reputation manager simply passes onwards information for storing; as connections to experience sources and external reputation systems are replaced by the simulator classes, it has nothing to analyze. As a result, the simulation setup does not include any policies for the Reputation manager to follow, either.

The system provides support for further simulations in the future as well. Experimentation with actual demo services passing SOAP messages is enabled by implementing the connection through the monitor plugins to the Pilarcos middleware prototype.

On the other hand, choosing a few existing representative reputation systems and implementing translator components for them makes it possible to demonstrate differences between the systems. In this sense, the system can provide an interoperability standard for reputation information content. We have designed the information model to be sufficient to support flexible policy adjustment through business concepts, and justified why a simpler model would not be able to cover the needed functionality.

5.3 Configuring the TuBE system

The TuBE trust management system can be adjusted to different decision contexts, configured to process reputation flows from a number of different reputation networks, and to separate routine decisions from those requiring human intervention. As the flipside of this power, the system is only as effective as the policies governing it. We have discussed example policies in Section 4.2 and will study their behaviour further through simulations and attacker analysis in Chapter 6. In this section, we take a look at where these configurations come from, which ones are necessary to specify and which ones can be covered by defaults, and how the configuration process can be otherwise simplified.

We begin by analyzing the minimal configuration needed. When provided with default policies for all policy points possible, TuBE matches a much more constrained fixed-policy system in ease of use at the user end. The system sets sane defaults wherever possible, and different features of the full decision-making can be separately configured to be used, rather than needing to configure them to not get in the way.

For a small enterprise operating with simple routine decisions, the main task of configuration can be delegated e.g. to the commercial provider of the trust management system. The power of configurability is in that the delegation can be revoked when and where necessary. Making configuration an easy task then becomes a question of designing the configuration tools to support these local exceptions from default behaviour, as well as using policies generated from other policy collections that are already set, such as the eContract governing the collaboration.

As a part of configuring the system, the point of triggering a trust decision must be decided. Default decision points for each service type can be defined by business network and service type modellers, however: The business process models define tasks which can be expected to form reasonable units of trust decisions, and the defined choreography can be tagged with information on when the risk-relevant parameters of a given task are known so that a decision can be made.

The only policies that cannot reasonably be covered by “factory-set” defaults are the basic principles of experience processing. There is no requirement to use both local and external reputation information if one half caters for all the needs of the enterprise, but both cases require some configuration:

- If local experiences are used, the system must at the minimum be told how to identify behaviour that is according to specification (i.e. a good outcome), which allows it to consider anything deviating from that to be negative. This approach resembles specification-based intrusion detection [Vil05a]. This specification does not need to be generated from scratch for trust decision purposes: it is provided by the business network model and the eContract governing the collaboration. The impacts of the outcomes, however, vary according to the size of the enterprise: whether, for example, the lost amount of money represents a minuscule part of the enterprise budget or constitutes a considerable financial problem.

- If external experiences are used, choosing an information source is necessary, but further analysis can even be outsourced to it: high-quality reputation information can be provided as a separate service for a cost, similarly to how credit rating companies operate today. For a truly minimal configuration, negative experiences may be reported manually rather than automatically shared; no sharing at all would undermine the social pressure that ensures the scalability of the service ecosystem. Standardization of reputation information formats allows the generation of ready translators for the reputation information, complete with mapping identities to the existing formats used for the service offers stored in eContracts.

In contrast, risk and importance can be fixed to be the same for all actions, context can be left unconfigured, risk tolerance can be based solely on whether negative experiences are present or not, and all outcomes can be reduced to minor positive or negative monetary effects. Precision is available for organizations that need it enough to invest into the configuration; returns of the investment improve as readily available standards and models allow semi-automated configuration and reduce repeat work.

Beyond fixed defaults, we find that the business network models produced by domain experts will form an important basis for extracting e.g. base risk values for specific actions, or for categorizing actions based on their business importance from the point of view of a successful collaboration. A domain expert is well suited to estimate the general risks and gains inherent for a given role in a collaboration; by embedding this information into the business network models, the configuration of base risk information does not require a separate modelling round within each enterprise using the model — unless their own observations disagree with the model. The connection between business network models and trust management system configuration remains optional to respect the actors' autonomy: the right to override all and any centrally determined policies and valuations.

While TuBE policies are rather simple to support their real-time enforcement in the Pilarcos monitor, we envision that these low-level monitor rules can be generated from a higher-level policy language. This will allow semi-automatic configuration changes through the TuBE configuration interfaces. Policy refinement and different types of policy languages have been discussed in more length in Section 3.3.2; for TuBE, we summarize the possibilities and impact of enhancing collaboration models with policy-relevant information in Section 6.1.

In conclusion, policies play a major role in the proposed trust management architecture, and therefore it is important to study their behaviour as well. To ensure that the administrative cost of configuring the system does not become too overwhelming, particularly for small enterprises, TuBE has been designed to operate reasonably in simple decision contexts using a minimalistic configuration, so that more powerful features of the model can be taken into use when needed. Synergy from other service ecosystem management tasks will ease the configuration of the trust management system specifically; policy refinement allows higher-level, more general policy languages to be semi-automatically translated into the more specific rules of TuBE.

5.4 Chapter summary

We have implemented the TuBE trust management system in order to evaluate the concepts and algorithms proposed in the previous chapters. This chapter has presented the internal structure, functionality and interfaces of the trust management system. The implemented core features support the central processes: the trust decision making, where reputation information is converted to a risk evaluation and compared to a risk tolerance policy, and reputation information processing, where incoming experiences are evaluated for credibility and introduced into the local reputation repository.

The TuBE trust management system operates in a distributed environment with other Pilarcos infrastructure services. The underlying Pilarcos infrastructure provides a large range of supporting information and components that TuBE is in a unique position to take advantage of. There is consequently no need to reinvent existing Pilarcos functionality, such as contract negotiations, within this work.

Central connections for the TuBE system include external reputation systems, which allow the system to learn from shared experience, and the Pilarcos monitor, through which the trust decisions direct the collaboration and gain input for local experiences. Due to the scope of our experiments in the next section, the simulation setup focuses on the two central internal processes within TuBE, and the policies involved.

Policies allow the system to adjust to different business situations. They are separated from the overall implementation, so that they can be changed at runtime. We have defined two central classes of policy: the trust decision policies, and the reputation update policies. Within these classes there are a number of more detailed policies, such as how to evaluate the credibility of a new experience.

We have also discussed how the behaviour of the system can be adjusted through policy configuration, as well as ways to ease the configuration task. The former is needed to allow the system to adapt to changes, while the latter topic is relevant to demonstrate the feasibility of the system in terms of the administration costs it incurs. Configuration tools can take advantage of the models made available by other infrastructure services in Pilarcos, such as business network models and the collaboration contracts. As a design choice, the configuration of local decision-making and reputation update processes is left in the control of local administrators at the service provider enterprise: if local policies and the shared contract come to be in conflict, the local policies are followed, although the cost of violating the contract is also considered as a business importance factor in the trust decisions. This respects enterprise autonomy.

The following chapter utilizes the simulation setup we have described for evaluation purposes. As a part of evaluating the attack resistance of the system, the behaviour and performance of a set of example policies is illustrated through simulations and a game-theoretic analysis of optimal attacks against them.

Chapter 6

Evaluation and experimentation

This chapter evaluates the TuBE trust management system through the system requirements set in Chapter 2, including adjustability to different business situations and attack resistance. The earlier chapters have specified an information model for multidimensional trust decisions, decision-making algorithms to allow private decisions at specific points of the collaboration, an aggregation algorithm for reputation to allow the system to learn from first-hand and shared experiences, and ways to privately adjust the system behaviour to different business situations.

In order to support the evaluation in Section 6.1, we present two experiments and game-theoretical policy analysis done on the implemented TuBE system prototype in Sections 6.2 and 6.3. The common simulation environment for the two experiments is described in Section 6.1.6. For both experiments, the first two subsections specify the fixed inputs. The first simulation experiment varies trust decision policies against different local experience streams, and the second fixes the trust decision policy to the best-performing policy of the first experiment, while varying reputation update policies against different external experience streams.

Section 6.4 presents related work on simulation experiments to demonstrate the state of the art in evaluating reputation-based trust management systems and contrast it to the methodology applied in the previous sections, while Section 6.5 extends this background into a discussion on the feasibility of creating a benchmark for trust management systems, which would provide a shared measure to help compare the performance of different proposals.

6.1 Evaluation criteria

In this section, we evaluate our proposed trust management architecture from the point of view of six system requirements: conceptual usability of decision making, support for autonomy, adjustability for different business situations, implementation of social control, scalability and feasibility, and attack resistance. For each criterion, we go through the set of proposed features that fulfil the requirements.

In addition to the evaluation presented here, the underlying Pilarcos architecture for open service ecosystems has been analyzed using a selection of methodologies for constructive research, such as architecture evaluation (e.g. SAAM and ATAM [CKK02, KBAW94, KKC00]) and creating prototypes for the different infrastructure services to gain experience on the feasibility of different approaches through them. Within the research group, expertise from different domains of computer science and software engineering is applied to study the potential failure points of the architecture as well as important scenarios to support both from the point of view of a single partner, and the entire service ecosystem. Interaction with companies through research projects also exposes the approaches taken to open criticism from the industry partners, while company use scenarios have provided cases and threat scenarios to address. A security threat analysis of the overall architecture [MRVK10, Vil11] has proved particularly relevant for TuBE.

6.1.1 Conceptual usability of decision making

The criterion for providing conceptually usable decision making tools underlines the necessity for using concepts that are consistent with the domain where policies are set. In other words, business-level concepts are considerably better for defining decision policies on business services than implementation-specific technical concepts. It therefore falls upon the trust management system to map business-level concepts to technical concepts that are implementable. Conceptual usability also encompasses the expressive power of the models.

The proposed trust decision model balances risks and incentives, defining the former in terms of past experiences and reputation, and the latter in terms of business importance and risk tolerance. All are influenced by context, expressed through the effects the business and technical environment has on the other factors. Risk, in turn, is not only expressed in terms of monetary assets being threatened, but also the reputation of the enterprise, its ability to control its own autonomy, and its satisfaction in its collaborations. All these aspects of trust decisions have been chosen with conceptual usability in mind.

In addition, the multi-dimensional model for trust and risk differentiates between small and large successes and failures, and their relation to subjective satisfaction of expectations as well as objectively measurable impacts. It can identify situations where no experience information is available, or where the information is too low quality to be credible. In related work, these two are often indistinguishable from situations where the overall experiences have been polarized but carry an equal number of positive and negative reports, or situations where experiences simply indicate that the known transactions had no actual effect either way. Our proposed models have the expressive power to adjust the reactions of the system to all of these cases to what is considered appropriate for each of them separately.

In terms of understandability, the basic models employ simple elements, steering clear from solutions that would detract from the overall usability of the system. While powerful statistical models, such as different types of probability distribution functions, can be applied in the decision policies when they are deemed necessary, they are not required to operate the system. Policies can be equally well be implemented based on simple additive rules, or even specific limits such as "no negative monetary experiences are tolerated in this case". Achieving the required expressive power for the models brings with it a necessary tradeoff in adding more elements for the user to learn, but the complexity resulting from the possibility of choice can be controlled through sane default policies that eliminate the effects of any unnecessary factors from decision-making.

In summary, the proposed trust management system provides support for expressing decision policies in terms of business concepts in a way that can differentiate between relevant situations surrounding the trust decisions. Using concepts familiar to the policy setter improves the overall usability of the system.

6.1.2 Support of autonomy

The criterion for supporting autonomy encompasses the need to allow services to make their own decisions, to be allowed to control their own information and to minimize the need for submitting all actors to centralized monitoring or rule enforcement.

The proposed trust management model is built on local trust decisions, made with a combination of private information and experiences shared through reputation systems between the members of the ecosystem. The trust decisions are based on balancing risk and incentives, both of which are subjective and therefore locally determined.

We distinguish between three types of information exposure: public, controlled sharing, and private. For trust management, shared reputation falls in the second category, and other fixed factors of the trust decision in the third category. Public information from business network models and shared information from collaboration contracts may be used as a reference in the process of defining these values, if the enterprise so chooses.

In accordance to the Pilarcos open service ecosystem model, all control is distributed to the service peers: local policies can override shared contracts, when the need is greater than the deterring effect of any compensation clauses in the contract. As with all collaboration contracts, the rules of reputation networks can also be overridden: the service can choose to not share experience over a collaboration or provide false information, and the surrounding environment is designed with this possibility in mind. In contrast, centralized monitoring would require an external party to place tamper-proof components within an enterprise's domain of control. While this results in more reliable data from the monitoring party's point of view, it disrupts the autonomy of the monitored party and creates an opening for uncontrolled leaking of information on e.g. what kinds of actors and business scenarios the enterprise is focusing on in its collaborations.

As another extension of the localized choice, we do not require that all members of the ecosystem must agree on a single reputation system to use. Instead, we allow reputation information to be collected from multiple sources, analyzed and combined into a whole that is suitable for a given enterprise. As a part of the VORe reputation system proposal [RK11], a specific collaboration contract can require sharing information in a given reputation system according to a given scoring rule if the participants agree on it beforehand, but even then the information is limited to that collaboration, and does not need to be used in local decisions if it is not considered sufficiently relevant or credible.

In summary, the proposed trust management system makes local decisions based on private, locally analyzed information, which can be complemented with external sources as needed. Centralized control is minimized; instead, different collaboration contracts contain the agreed-upon rules, and monitoring is done by each service over the part of the collaboration it can observe without invading the domains of control of other participants. In addition, information flowing out of the service is controlled primarily by local policies. These factors ensure that the system supports enterprise and service autonomy.

6.1.3 Adjustability for different business situations

The criterion for adjustability for different business situations connects to the architectural requirement for providing ways to adjust the trust decisions. Under this criterion, we analyze whether these ways provide sufficient support for the system to evolve gracefully as the needs of the organization change. This evolution can take place within two processes: the collaboration process, and the design and ecosystem evolution process.

During the collaboration process, TuBE automatically adjusts to different situations through learning from experience and supporting metapolicies for determining when a decision is, in fact, routine enough to be automated. Learning from experience automatically is a significant improvement to certification-based trust management, which relies on manual revocation processes. In addition, the architecture separates data aggregation policy from decision-making policy, which allows it to collect experiences once to use them in different decision contexts, without losing information. This design choice is coupled with reputation epochs to ensure that the system can react to changes in reputation flows while following the same separation: the change is noted when data is collected, but its weight is only determined at decision-time.

The TuBE information model provides input to the metapolicy that determines when a decision is a routine case suitable for automation. Central measures for such a decision reflect the amount and quality of information available to base the decision on, as well as the magnitude of risks related to the decision. The amount of quality of information is measured in terms of the number of experiences collected, the credibility of the experience data, and the consistency of the actor's behaviour measured as the number of reputation epochs the experiences are divided into. The risks related to the decision can be measured both as specific probabilities of e.g. high monetary losses, but also comparing the probability of minor effects and major effects respectively. With the help of a context filter affecting the risk tolerance categories, a given action can be set to be made as a manual or automated decision independent of the other data available as well.

As a part of the ecosystem evolution process, the needs of the service provider enterprises and the ecosystem as a whole may change. Manual policy configuration can be used to reflect the different and evolving roles of actions, actors and contracts in policy, to adjust to temporary changes, and to consider the evolution of the entire service ecosystem.

From the point of view of a single enterprise, for example, the action of buying or selling a product requires a different analysis of risk than providing a price list to a requester. A long-term strategic partner may require a different treatment in decisions than a new partner, which, in turn, can either be evaluated for suitability through an opportunistic experiment of low-risk activities, or as a highly desirable addition to a strategic network that is primarily evaluating the trustor instead.

Similarly, the importance of contracts differs: the incentive to continue a collaboration even at a short-term cost is higher if there are foreseeable future gains to be made to balance for it. Temporary changes can include e.g. preference adjustment to a need to move products in order to free up storage space and generate cash flow, more conservative risk evaluation due to an observed ongoing attack taking place in the ecosystem, or even declining new requests to create new products when the collaboration is about to move to its termination phase.

We envision that in the future, TuBE rules can be generated from a higher-level policy language, which allows semi-automatic configuration changes. Policy refinement makes it possible to set up strategic policy configurations that affect trust management policies as well as e.g. privacy policies, or the sets of acceptable business network models.

The entire ecosystem may face pressure to evolve when new ways of making business are discovered: for example the emergence of cloud services for information storage and virtual, load-balanced servers is dropping the hardware, setup and maintenance investment needed for a small actor to offer high-availability web services. Other services and products may become unsellable: for example setting up a wiki used to require server space and someone to install and maintain the software, while consumer-grade wiki platforms are now offered essentially free of charge[1]; similarly, email, calendar and scheduling for private users as well as online encyclopedias, travel guides and even some travel agency services must currently compete with free alternatives[2]. The trust management system must be able to adjust to changes in the ways business is made: the sets of interesting actions can change, the business network models be replaced with new ones when collaboration needs evolve, and new relevant reputation networks may emerge.

The TuBE trust information model is multidimensional, including rep-

[1]For example Wikia (`http://www.wikia.com/`) hosts a number of wikis for e.g. interest groups and collaborative software projects.

[2]For example Google (`http://www.google.com/`) offers email and a shareable calendar service, while Doodle (`http://www.doodle.ch/`) offers a more specialized scheduling and poll service. Wikipedia (`http://www.wikipedia.org/`) is a community-driven multilingual encyclopedia, and Wikitravel (`http://wikitravel.org/`) a travel guide with a similar operational model. In turn, online flight search services like Amadeus (`http://www.amadeus.net/`) and centralized hotel booking systems have enabled consumers to essentially bypass travel agencies in organizing trips abroad.

resentations for both risk and incentive. The former is important in evaluating what the expected losses or gains of an action are based on the future actions of the trustee, but it is equally important to model that a trust decision depends on the trustor's state: they may have reasons independent of the trustee's behaviour to accept or deny a request. Further rule refining can be made through context filters, which allow setting up special cases that depend on e.g. the given contract and time period within which the request is made.

As a part of the larger whole of the Pilarcos collaboration management middleware, TuBE produces automated trust decisions in routine situations, and focuses on whether the activity is worth the risk. Other parts of the infrastructure cover the evaluation of whether the collaboration is interesting in terms of belonging to a suitable business domain, and ensuring that a proposed contract or action is not inherently clashing with e.g. local privacy policies. This separation of concerns in decision making allows policy adjustments to be made on the level that is most relevant to the change at hand.

The Pilarcos open service ecosystem supports ecosystem-wide evolution as well, including the addition of new actors and new types of collaborations. Traditional strategic networks with predetermined partners can be used when they are considered to be the most beneficial, while the trust management system supports opportunistic experimentation on new collaborations and partners as well. In terms of collecting reputation information on these new partners, TuBE is not limited to a single external reputation system, but is designed to connect to multiple systems of different types, which makes it possible to actively adjust where there is a lack of experience information on any given actor.

In summary, as the business environment and the enterprise evolves, the decision-making infrastructure must evolve with it. TuBE and the underlying Pilarcos architecture can adjust to different situations through the combination of a sufficiently expressive information model providing support for a range of policy configurations, and an architecture design to provide automated reactions to changes in behaviour and available information.

6.1.4 Implementation of social control

The criterion for implementing social control in the open service ecosystem demands that in the absence of a central authority to enforce rules, contract violations and other misbehaviour are sanctioned in a distributed way. In order for this to work, the sanctioning system must operate on two levels:

punishing regular, i.e. first-order misbehaviour as well as sanctioning any unfair punishments (including lack of deserved punishment) to first-order misbehaviour, on the second level. This combination allows the community represented by the open service ecosystem to remain operational even as it grows in size.

In reputation-based trust management, behaviour control is primarily based on reputation. Misbehaviour is punished by a drop in reputation. Reputation itself is simply a measure, but when reputation information is effectively disseminated and persists over time, it influences the business opportunities available to the members of the reputation network. Good reputation increases the desirability of the service, while bad reputation deters potential collaborators. If the reputation of a service drops very low, it is effectively shut out of the ecosystem — other services do not end up choosing it as a partner.

The effective dissemination of reputation information consists of four factors: well-defined semantics for reputation, motivation to share information in the system, support for credibility analysis of the reputation information, and controlling the leverage that any single actor has on the reputation of others. Most of these aspects are determined by the reputation system in use, and TuBE does allow different kinds of reputation systems to be used for trust decisions. Our analysis will focus on the VORe (Verifiable and Objective Reputation) system proposed in Section 4.1.4, as it has been designed to fulfil the requirements of inter-enterprise collaboration in particular.

In the VORe reputation system, the semantics of shared reputation is agreed upon in contracts and business network models: a specific type of contractual outcome translates to a specific kind of experience shared in the reputation system. This makes reputation less contextual and increases its fairness, ensuring that for example different expectations do not cause different experience reports for the exact same behaviour. The motivation to share information is also ensured through making reputation sharing a part of the contracts that are negotiated and agreed upon before collaboration. Failure to report or a misreport are punished by a reputation loss, the extent of which is agreed upon in the contract for the reputation network itself. Misbehaviour in reporting is, in other terms, a contract breach like any others. Fear of retribution, a central reason to not report negative experiences, is strongly reduced when the outcomes are well-defined and the reports required in the contract. Reciprocal experiences, i.e. positive reports given in return for positive reports, and negative for negative, are limited by requiring well-defined evidence to back them up.

In VORe as well as other reputation systems, credibility analysis remains essential. Despite the audit trail of evidence built into the VORe reputation system, colluders can still claim to have transacted without actually doing anything but producing the receipts. On the other hand, presenting false evidence about honest actors is a verifiable breach of contract. Contractual breaches are more solid cases for legal repercussions than a typical case of spreading of misinformation, unless it is extensive enough to be judged as libel. No matter what reputation systems it connects to, the TuBE trust management system supports local credibility analysis to be made in addition to any analysis that is defined within the reputation system itself.

Limiting the leverage of any single actor on the reputation of others encompasses two design choices: First, the cost of creating a new identity must be balanced with the power it brings in the reputation system. While creating new services is straightforward, the number of identities they translate to in a reputation system can be tied to the identity of the enterprise to limit their influence. Second, to avoid amplifying single experiences through repetition, only first-hand experiences should be shared in the reputation network. TuBE ensures this through storing local experiences separately from external reputation information.

To support the persistence of reputation information, it must be tied to a persistent identity. While assigning such identities to private people is a problem commonly observed in related work, enterprises must necessarily have a persistent identity bound to its legal entity in order to sign electronic contracts. In other words, persistent identities are ensured by the service ecosystem.

In further support of this goal, the proposed internal reputation information model ensures that no information is lost, and no old transgressions or successes forgotten. While policies may be changed to give more weight to recent behaviour than to old, aggregated experiences, they are separated from the information gathering policies in order to ensure that the weighting can be changed later. Reputation information is collected once, then used in multiple different decision contexts.

As a final factor of social control, reputation epochs ensure that an amassed good reputation does not outweigh recent transgressions. The calibration of exactly how much patience is extended towards previously well-behaved actors is configurable through trust decision policies.

In summary, the proposed trust management and reputation management systems implement social control in the service ecosystem by creating incentives to behave well and gain reputation, while failure to follow shared rules is punished by reputation loss.

6.1.5 Scalability and feasibility

The criterion for scalability and feasibility of the systems implementing the proposed architecture encompasses the requirement that the costs and benefits of implementing such a system must be in balance. Costs can be divided into two categories:

- computational costs encompass how heavy the system is to operate in terms of algorithmic complexity, messaging and storage, while
- administrative costs comprise the necessary human labour to set up and operate the system.

The computational costs of the trust management system accumulate from three processes: first, making a trust decision, which must be done in real time; second, processing monitor information into experiences and passing it into reputation systems; and third, processing incoming reputation information. The algorithmic complexity of the system is dominated by searching for information from ordered databases, such as the local reputation data storage; with e.g. a binary search, such information retrieval grows logarithmically in relation to the number of elements stored in the database. The messaging complexity within the system is low, while the second and third processes involve communication with a large external reputation network; we will analyze it further in relation to these processes. We summarize the storage requirements of the overall system at the end of this analysis. Refer back to Figure 5.3 for an overview of the architecture.

Table 6.1 summarizes the different activities in the three processes and gives an estimate of the scale and type of the relevant variables involved. Search operations are assumed to be made to ordered databases and therefore logarithmic in complexity, while the processing of rules and contacting different actors in a reputation network grows linearly with the number of items to process.

The *trust decision process* consists of identifying the need for a trust decision, retrieving the supporting information based on the action and trustee involved from local storage, and processing it into a decision. The first phase is done based on the type of the incoming message sent to the business service and the relevant contract identifier. It involves a straightforward local lookup on whether the necessary decision has already been made for this contract. We assume that messages triggering trust decisions arrive to the trust management system in correct order, as other monitoring facilities follow this process for reasons of keeping the state of the shared business process up to date. Locating the base risk for an action can be

Table 6.1: The algorithmic complexity of the processes in TuBE.

Activity	Complexity	Operation and expected scale
Trust decision process (real-time)		
Identify message type, extract parameters	O(r_t)	process: 10 rules
Check for existing decision	O(log t)	search: 10^2 transactions
Find relevant action data	O(log r_a)	search: 10 formulae
Find reputation for trustee	O(log n)	search: 10^6 actors
Process open experiences	O(e_o)	process: 10 objects
Merge reputation epochs	O(e_e)	process: 10 relevant epochs
Find, apply context data	O(c log c)	search and process: 10 filters
Apply risk tolerance policy	O(1)	process: 1 policy
Delegation to human	-	(No longer real-time)
Processing monitor information (background process)		
A message into events	O(r_m)	process: 10 rules
An event into experiences	O(r_e)	process: 10 rules
Store experience locally	O(log n)	search: 10^6 actors
Send to reputation network	O(1)–O(n)	process: 10^6 actors
Receiving reputation information (background process)		
Receive updates	O(1)–O(n)	process: 10^6 actors
Analyze credibility	O(1)	process: 1 policy

considered a low, constant cost, as the number of actions monitored by a single trust management system instance is assumed to be small.

The second phase of the trust decision process involves locating the trustee's experience information from the local reputation data storage. The cost of this lookup from an ordered database grows logarithmically with the number of actors whose reputation information is stored in the local repository. Including any open experiences within the reputation counters is a linear task; the number of open experiences is expected to be small. If reputation epochs are used in the decision and not pruned to limit their number, calculating the risk and comparing it to a risk tolerance policy like the ones considered in this thesis can grow linearly with the number of reputation epochs stored for the given actor. This assumes that each epoch would be added to the result with a separate weight, for example.

The cost of the third phase of trust decisions depends mainly on the number of context filters that are relevant for the case. With the example policies considered in this thesis, and the assumption that the number of active context filters is also reasonably low, the costs of this phase can be approximated to be constant: the processing of context filters consists of comparisons and basic arithmetic, and the context storage database can be suitably indexed to make finding relevant context filters a fast operation. In general, the cost of processing context filters therefore grows linearly with the number of relevant filters for each case. The final comparison to a risk tolerance policy is a constant cost, given that the number of tolerance categories is low.

We conclude that real-time trust decisions are computationally feasible, as the most dominant algorithmic cost is involved with the logarithmic complexity of locally looking up experience data for a given actor. This cost is quite low; the complexity is comparable to that of e.g. a regular DNS lookup, which forms the core of the Internet naming system: its complexity grows logarithmically depending on the number of actors as well. As with DNS, the necessary data can also be cached close to the monitor instances needing it, in case the total number of actors known to the enterprise grows sufficiently large to slow down decision-making.

This analysis only covers automated decisions. When the trust decision is found to fall within the gray area that requires human intervention, the result by necessity becomes delayed. Human decisions should only be invoked in locations of the process where there is at least a day's time to react, unless the process is constantly monitored by specialized staff around the clock. Until the human decision process is finished, the essential effect of the gray-area categorization is a negative decision. In other words, if the contract specifies a timeout that is passed before the answer arrives, the service has been denied as far as other collaborators are concerned.

Processing monitor information into experiences is done as a background process. Its computational complexity depends on the number of relevant monitor events per task, and the different rules that a single monitor event can invoke. We assume that for a given type of collaboration, the number of different events and rules to process is quite low; again, we expect to benefit from pre-processed and therefore conceptually higher-level monitor events that can be produced as a side effect of tracking the state of the shared business process for other purposes.

Monitor rules dealing with experience production consist of comparisons and simple arithmetic done on the message parameters, and cross-checking the outcomes with other messages within the same task (i.e. the activity producing a single experience). The number of different rules invoked by a given message type is expected to be reasonably low; as with context filters, the complexity of processing these rules grows linearly with their number. Each service instance should run its own monitor process, which performs the bulk of the analysis; only the experience updates are centralized to a single experience store. This reduces the threat of experience processing forming a computational bottleneck for all the services of an enterprise as well. While Pilarcos monitor rules can depend on the outputs of other monitor rules that provide shared preprocessing, they are not allowed to form cyclic dependencies; this limitation allows further performance optimization [HKL+04].

The messaging cost of passing the generated local experiences into an external reputation network depends strongly on the dissemination method of the network. The upper limit, for a completely unstructured reputation network, is equivalent to the reporter having to send a message to every member of the reputation network for each reported experience, i.e. it grows linearly in relation to the number of members in the network. The minimum cost to the reporter itself is a single message sent either to a centralized repository or to a connection point which is responsible for disseminating the information. The real cost is somewhere in between, shared between the members of the reputation system. In a reputation network similar to the VORe reputation system discussed in Section 4.1.4, all network members interested in experiences on a given actor must receive the experience information at some point, one way or the other. The number of total messages sent for a given experience is therefore dependent on the number of reputation network members, and may be at most tripled if the experience is rebutted. The dissemination process can be made more efficient particularly when there are trusted third parties present who can cache and disseminate reputation information for multiple participants, essentially representing them in the reputation network.

The computational cost of *receiving reputation information from external reputation networks* is similarly dependent on the dissemination method of the network: where the maximum cost for sending out experiences is present when the reporter has to actively push the information to all other participants, the maximum cost for receiving is reached if the recipient must query for, i.e. pull, the information from all other participants. The push method is the more efficient of the two from the point of view of the entire reputation network, as it avoids the additional messaging load from unnecessary polling for new experiences, and has the additional benefit that interest in a specific actor's reputation is not shown actively to other members of the network. A centralized trusted server storing reputation information can employ the publish/subscribe paradigm to ensure that new reputation information is pushed swiftly to interested parties and only them. In contrast, distributed peer-to-peer dissemination carries some additional messaging or storage costs for maintaining an overview on subscribers, and the interest information indicated by subscriptions may be too sensitive to share with the entire network; as a result, peers may opt to subscribe to everything to maintain their privacy.

The transformation and other processing per experience forms a constant cost both in sending out and receiving reputation information. A central difference between reporting and receiving of reputation information is that the receiving is done only once for the local reputation information repository: a single repository can cover multiple services within an enterprise, and it definitely serves multiple service instances.

In summary, the algorithmic complexity and messaging costs of reputation management are comparable to those of general information dissemination in distributed systems: there are tradeoffs between centralized and distributed approaches, but in itself TuBE does not introduce particularly complex processing or messaging needs.

For storage costs, the largest data structure needed for supporting the trust management system is the local reputation data storage. The base experience information is stored in a constant-sized set of counters, and the total storage required depends linearly from the number of actors whose reputation is tracked; not all actors in every reputation network are necessarily interesting in this sense. Reputation epochs should be regularly pruned for real-time use; storing hundreds of epochs per actor does not notably increase the quality of decisions compared to storing the total number of epochs and the most relevant handful of latest changes.

If storage space is not an issue, a trail of every experience ever stored can be kept in a separate storage and used for data mining. The condensed data structure we have discussed above is essential for real-time decisions, while the full uncompressed experience data could be useful for example to automatically discover more effective rules for reputation epoch changes. This kind of processing is separate from the proposed trust management system; instead, it may support system configuration, essentially trading automated processing for the time spent by human users performing the policy design and configuration.

To conclude, the computational costs of the trust management system, in terms of algorithmic complexity, messaging and storage costs, are reasonable and ensure the scalability of the system. A dominant feature in the cost is the number of actors tracked in the reputation system, as the number can both expected to be large and affects messaging costs as well as local searching costs, to a lesser degree. The number of different policies applied by the different enforcement points, such as the number of active monitor rules and context filters, is another factor that linearly increases the cost of decision-making; we consider these costs to be more controllable, however. Processing-intensive elements in particular are replicated in the system to limit the formation of bottlenecks in the automation. To conclude the analysis of the feasibility of the proposal, we will now consider the administrative costs of setting up and running the system.

The administrative costs of the TuBE trust management system should be considered in the context of the entire Pilarcos collaboration management infrastructure: its different functionalities support each other, and while the cost of configuring a supporting subsystem may be high, the cost of running the system without it could be entirely unfeasible. We will therefore now focus on the maintenance costs of the proposed TuBE system as a part of Pilarcos. Aspects of the configuration of the TuBE policies specifically have been discussed earlier in Section 5.3.

The maintenance costs of the trust management system as well as the open service ecosystem in general fall to two categories: the initial investment needed to set up the system, and the administrative costs of its operation.

The initial investment for adopting the overall Pilarcos infrastructure is high: it requires enterprise architecture changes and personnel education in the new orientation towards services and models. At the same time, applying the infrastructure promises significant cost reductions later on, due to making change management depend on reconfiguration, i.e. updating metainformation and models, rather than re-implementation [Kut02].

The overall administrative costs for supporting the automation of collaboration management tasks have also been suspected to be very high [RK10]. Business network models, service types and service offers must be produced and published in the respective information repositories. Further, trust information must be annotated into relevant models of e.g. inter-enterprise collaboration and enterprise risk management in order to make the information available to automated processing.

On the other hand, these activities are a part of the normal design and management processes in enterprises. On these areas, the Pilarcos architecture provides an opportunity to move from low-level, technology-dependent expressions to higher-level conceptual work at the business level. This is reflected in conceptual usability as well: policy-setters use high-level policy languages to express their goals, which are then refined to automatically enforced lower-level policies.

Producing the base action risk information relevant to trust management can be pushed to domain experts. Business network modelling must be done for contracting and collaboration enactment purposes, and analyzing the risks for any enterprise activities can be based on the same models. This combination makes it possible to enrich trust decisions with the results of a single modelling effort, rather than repeating the analysis in each enterprise separately.

In summary, the trust management system and the Pilarcos collaboration management system both require an administrative investment in order to automate processes such as routine decisions, interoperability management, information processing and monitoring. We argue that the investments are worth the gains from automation: a flexible system requires more configuration than a rigid solution, but we expect that in the context of inter-enterprise collaboration, a rigid solution will very quickly turn out to be outdated and therefore unusable.

The proposed trust management system brings opportunities for automatically adapting to changing business situations. The key strengths of the open service ecosystem approach lie in its openness and flexibility: the ecosystem is open both to new business models and to new actors, and it is scalable. There is no single reputation information source but multiple, and each enterprise can choose which reputation source to connect to.

To balance for the costs, the trust management system allows opportunistic behaviour and experimenting on collaborations with new partners made available in the ecosystem. On the other hand, it allows collaborations to be set up based on traditional strategic networks as well; policies approving only specific well-known partners are by no means prohibited. The ease of setting up new collaborations supports both small and medium enterprises in reacting to business opportunities that they could not cover alone, and large enterprises in exploiting new niche markets.

6.1.6 Attack resistance

As all computer security, our criterion for resisting attacks reflects a balancing between two goals. On one hand, the trust management system should be resistant to and able to punish different forms of misbehaviour, including coordinated attacks. On the other hand, it should remain usable and not be so strict that would severely inhibit business in comparison to human decision-making. In policy-based systems such as TuBE, the balance can be adjusted through policy configurations, as long as the underlying information model supports the level of expressiveness needed. We will first present the elements in the trust management system designed for addressing different known vulnerabilities, and then discuss our experimentation on different policy configurations.

We have set out to address eight types of vulnerabilities observed in different existing reputation-based trust management systems in our survey [YRX12]:

1. Inability to differentiate between situations where a routine decision can be made with sufficient information, and different forms of problematic decisions.

2. Not differentiating between high-value and low-value transactions in reputation, enabling value-based attacks where low-cost transactions are used to boost reputation and high-cost transactions to maximize profit from misbehaviour.

3. Making it too difficult to differentiate between qualitatively different threats, e.g. forcing policies to set a monetary price for the autonomy of the enterprise.

4. Having insufficient information available for decisions due to sparsity.

5. Inability to detect and punish misinformation in reputation sharing, enabling a plethora of reputation attacks aiming at defamation of competitors or whitewashing of colluding attackers.

6. Imbalance between the cost of creating new identities and the leverage gained through them in reputation sharing, which manifests through Sybil attacks [Dou02] and other forms of ballot stuffing.

7. Reacting slowly to changes in behaviour, which provides a large window for profiting from gained reputation through misbehaviour.

8. Completely forgetting old transgressions after a while (i.e. time-based decay of reputation information): the time to forget grows shorter when reaction speed is increased.

The first four vulnerabilities relate to the decision-making model, while the latter four focus on the reputation information.

The first vulnerability is introduced the moment we set out to automate decision-making: routine decisions are suitable for machines, but there is a number of ways the situation can become problematic. Our first requirement has therefore been that non-routine decisions should be delegated to human users, which has created a need to distinguish between the two types of decisions. The information model provides means to set up this kind of delegation based on particularly high stakes (risk), insufficient information to support the decision (amount of reputation information), insufficient credibility of information (measure of credibility of reputation information), the trustee's unpredictable behaviour (number of reputation epochs), or special cases based on a given actor, contract, action or time (context filter triggers). All these factors can also be used to provide an automatic negative decision, for example.

Differentiating between high-value and low-value transactions becomes relevant in inter-enterprise collaboration, where the stake and business value of different actions can vary greatly. As all experiences and decisions cannot be treated equally, we have introduced five different levels of impact for our risk and reputation values: major negative, minor negative, minor positive, major positive and no effect. With this support, risk tolerance policies can be adjusted to e.g. allow even somewhat probable minor losses to gain experience on a new collaborator, as long as there is no risk of major losses.

The risks of inter-enterprise collaborations are not only monetary, and we find that this should be reflected in the risk information model. A common, if somewhat mundane example of evaluating different risks in decision-making involves trying to determine whether a person who is known to be good at fixing cars would be a reliable babysitter, as the two tasks and their related risks are considered very different. From our point of view, competence is not the central issue here: offering a service of a given type is a statement that the actor is competent in the field. The main problem is the different risk taken (the functioning of a car versus the wellbeing of offspring), and whether the two can be easily captured by some common measurement. We find that for opportunistically exploring collaboration partners in the open service ecosystem, it may pay off to accept some monetary losses occasionally, while observed security problems should be taken more seriously. Negative experiences on the latter, in turn, can be a show-stopper for some kinds of actions, while not an issue at all for others. In order to upkeep a reasonable selection of collaboration partners, it is important to be able to tolerate past problems selectively: our asset-based and impact-aware risk model is at the core of implementing this goal.

When the differentiation power of the information model is increased, we must be careful not to introduce chronic sparsity in reputation information, i.e. not having the right type of information available for different types of decisions. We have introduced the four-asset model as a compromise between setting experiences for every type of activity separately and not being able to distinguish between different kinds of threats at the other extreme. Policies, in turn, can be used to convert one type of information to another (e.g. to sum together high-value and low-value transactions with a coefficient, or to convert bad reputation to monetary terms) at decision point rather than losing the particularity of the data by storing it pre-converted.

The first of the reputation-related vulnerabilities reflects the value that reputation information gains when decisions are based on it: the measure introduced to deter attacks becomes an attractive target of attack itself. To counter these attacks, misinformation must be identified and punished in reputation sharing. In order to punish misinformation fairly, the truthfulness of a given experience must be well-defined; this is not the case in most reputation systems, which are based on more or less subjective feedback. We have proposed a reputation system based on objective and verifiable experiences to allow us to detect and punish false reputation reports. On the other hand, local credibility evaluation is introduced both to deal with input from subjective reputation systems and to notice when shared reputation information is simply not reliable as a predictor of an actor's future behaviour, for example due to discrimination attacks. We will return to how credibility information can be used in trust decisions in Section 6.3.

Balancing between the cost of creating new identities and the leverage gained through them is an issue that particularly plagues systems with low cost of entry, such as electronic marketplaces and peer-to-peer systems directed at private consumers. For an open service ecosystem, the cost of entry is not particularly high, but it involves setting up the means to sign collaboration contracts that are legally binding. This requirement fixes the cost of creating new identities: it is equal to creating a new legal entity, such as a company; discarding these identities is therefore also somewhat more regulated than getting rid of one's pseudonymous login for a random Internet service. Through this contract-signing identity, connecting separate business service instances to their host enterprise (or a similar relevant entity) is straightforward, and allows reputation systems to limit the impact a single organization can overall have on the reputations of other services. In addition, ballot stuffing attacks are less attractive because of the lack of globally agreed-upon reputation information: each trustor privately evaluates what reputation information they find credible, and recommenders whose input does not support the trustor's predictions are discarded.

Slow reactions to changes in behaviour are a difficult problem for reputation systems. On one hand, old reputation information is valuable in placing any recent problems in a larger context, while relying too much on the past makes attacks based on changing behaviour attractive. We have identified this problem as another point where balancing between two goals is necessary, and therefore it must be adjustable through policy. Reputation epochs make it possible to give more weight to recent behaviour as opposed to old, and to vary the weight based on the decision at hand. The number of reputation epoch changes observed for a given actor also acts as a measure of the unpredictability of their behaviour; if the number of epoch changes is high, a human user should be involved to analyze whether the actor forms a threat or simply does not fit into the epoch categorization in use.

Completely forgetting old transgressions in time-based discounting of reputation information is related to the previous vulnerability: context is lost when old data is discarded. Through the reputation epoch approach, we do not lose any information; pruning reputation epochs does not forget old experiences, it simply clumps e.g. sufficiently old experiences together into groups within which all experiences are given the same weight.

In summary, TuBE both contains elements that directly deter attacks, and provides extensive support for policy configuration to locate the balance between too strict and too lax protection against attacks. The overall behaviour of TuBE depends strongly on the policy configuration in use. More specifically, while a poor reputation can be expected to lead to negative trust decisions and good reputation to positive, the actual definition of "good" and "poor" and the specific effect a trustee's reputation has on trust decisions depends directly on two central policies: the trust decision policy, and the reputation update policy, which incorporates new experiences into the trustor's reputation view. In order to understand the impact policies have on the resulting decision, we must study their behaviour.

In the following sections, we present the results of two simulation experiments to illustrate the effect of policies on the operation on the trust management system, and to identify desirable characteristics of the policy types being compared. The example policies have been chosen to cover central aims within decision-making and reputation updating, and they demonstrate some of the strengths of the TuBE trust management system in comparison to current state of the art.

For the purpose of the simulation experiments, the core TuBE system as described in Section 5.1 has been implemented in Java and run on the Java virtual machine (version 1.6), and the simulation tests run on an Linux box (Ubuntu Lucid) running on Intel Core2 Duo CPUs and 3GB of RAM, based on fixed inputs generated through rules such as "3 major positive experiences, one major negative", which leads to every fourth experience (exactly) indicating a major negative impact on the monetary asset. No random data was generated for the purpose of these experiments, as we do not use statistical models in this experiment; instead, the measurements are based on example experience streams and streams purposefully designed to bring out the worst performance in the tested policies.

The policies and simulated experience streams used have been created for the purposes of this thesis and presented in Sections 6.2 and 6.3. For measuring resistance against rational attackers, we analyze optimal attacks by hand with game-theoretical means. Simulation-based benchmarks are a poor tool for analyzing attackers [Gol11], while they can be prove useful in measuring the capability of the system to allow specific desired forms of behaviour. Real or realistic inter-enterprise collaboration experience data is not readily available, either. We contrast this to the state of the art on simulation and experimentation within trust and reputation management systems further in Section 6.4, and discuss the possibilities and limitations of benchmarks further in Section 6.5.

6.2 Experiment 1: Comparison of decision policies

In the first experiment, we compare the effects that six trust decision policies have on simulated collaborations. The policies are selected to provide a representable set of decision policy types, and we observe how effectively they discriminate against actors whose behaviour is not beneficial, as opposed to those who perform well. There is a tradeoff between not allowing any failures or misbehaviour, and keeping the set of possible partners from being needlessly narrowed down; we return to this challenge in Section 6.2.4. The best-performing policy of this experiment is used as trust decision policy in the second experiment described in Section 6.3, which then varies the reputation credibility factor while evaluating different reputation update policies instead.

We use two methods to study the effects of the policies: on one hand, we apply each decision policy to a set of differently developing reputations and observe their behaviour, and on the other hand, we study how well an attacker optimized against the decision policy would be able to perform. A synthesis of the findings is presented at the end of the section.

6.2.1 Decision policies

We first briefly define the fixed and changing variables of the experiment setting, and then specify the trust decision policies to be compared.

In TuBE, the trust decision policy is influenced by multiple factors; we focus on capturing the interaction between a given stream of experience and a given risk tolerance formula. To emphasize this connection and to control the scope of the experiment, we fix the actor, action, its importance and the decision context to not have any additional effect on the decision.

The trust management system is given a stream of local experiences of given values, and after each experience is incorporated, the gathered reputation is used as a basis of a trust decision by transforming it to risk and comparing it to the given risk tolerance formula. While the main point of interest is in whether the trust decision is positive or not, the tolerance formulae also provide us with a scoring interpretation of the information, i.e. how strongly positive or negative the decision is.

Each of the six risk tolerance formulae is applied to seven different experience streams. At the start of each stream, the reputation of the actor is purged to start the experience gathering from scratch.

There are three important concerns related to selecting a tolerance evaluation policy. First, since reputation is used as a medium of punishing misbehaviour, good reputation must be slowly gained but quickly lost. Second, as all transactions do not require equal investment, experiences from low-value actions should preferably be separated from high-value ones when making decisions as well: in eBay terms, actors should not be able to sell a few toothpicks to gain a reputation as a trustworthy car seller. Finally, differentiating between experiences should not lead to information sparsity, i.e. lack of suitable information for making a well-founded decision.

While the six different tolerance formulae have been chosen to illustrate the central concerns above, they can also be found in slightly different formulations in related work. For example, the policies also represent basic classes of trust decision policies, which can be deduced from the trust score calculations summarized by for example Jøsang et al. [JIB07].

The six risk tolerance formulae are as follows:

A: Basic: A basic "must not have had more negative experiences than positive experiences" policy: no difference is made between minor and major effects. This policy provides a baseline for comparisons.

B: Pessimistic: A more strict policy, "must have had 3 positive experiences per each negative experience". Again, no difference is made between minor and major effects. This policy reflects the need for quickly losing reputation due to misbehaviour: three good experiences are needed to cover for each negative experience. The multiplier has been chosen to illustrate the difference already with a small number of experiences.

C: Sharp: Like A, but with triple weight given to experiences with major positive or negative effects, as opposed to minor effects. This policy is a tradeoff between the second and third concern: it avoids information sparsity by transforming experiences from one class to the other with a multiplier.

D: Sharp-pessimistic: Like B, but with triple weight given to experiences with major positive or negative effects, as opposed to minor effects. This policy reflects the first concern as well as balancing between the second and third concerns.

E: Separative: Separating minor and major experiences to their own classes: the number of major positive experiences must be at least equal to the number of major negative experiences, and the same for minor positive and minor negative experiences. The different effect classes are not translated to each other. This policy reflects the second concern of differentiating between high- and low-value transactions more strongly than C.

F: Separative-pessimistic: Like E, but with triple weight given to negative experiences. This policy combines the first two concerns.

The *Basic* policy A corresponds to decision policies based on rating averages, which treat all experiences as equal [JIB07]. The main underlying assumption is that actors mostly behave either well or badly, and do not fluctuate between the two. Any stray experiences opposite to the general trend are considered more a measurement error than calculative changes in behaviour. The *Pessimistic* policy B, on the other hand, represents the approach of weighted averages [JIB07]. The weights are generally chosen to emphasize negative experiences over positive in order to punish misbehaviour more severely. This policy can also be seen as a more clear attempt at giving the trustee motivation to behave well through the threat of punishment, while policy A is simply observing the reality.

The *Sharp* policy C represents a moderate approach to the second concern of separating experiences on low-value actions from high-value ones. The TuBE trust information model allows giving more weight to the latter and slow down reputation gains from minor actions. As a result, the actor must do more small transactions in order to have the reputation equivalent to a single large transaction. The underlying logic is that once enough small transactions pile up, they eventually amount to the same benefit to the trustor as a single large transaction. For example, a seller could consider two partners to have invested into the business relationship equally much, when one is a long-term customer who places regular small orders, and the other is a new customer who has placed a few large orders.

The *Separative* policy E represents an even stricter approach to the same concern, keeping small and large activities completely separate. The policy reflects situations where transactions defined as large require investments that only relatively few actors are willing to make, and no others should be trusted to handle them, despite their good track record on other activities. In comparison, small transactions are something almost any service provider can be counted on, and it becomes more attractive to extend the competition to more providers for these tasks in the hopes of a better deal. This principle applies also with private people: we generally do not buy a car from a random marketplace vendor, even though we do not mind using the same amount of money to buy smaller items from a set of equally random marketplace vendors over a period of time — we are more selective of trading partners when larger amounts of money are at stake at once. Finally, the combination policies *Sharp-pessimistic* (D) and *Separative-pessimistic* (F) modify C and E respectively to comply with the stricter philosophy that good reputation should be quickly lost when negative behaviour is observed.

The policies above are expressed in terms of the number of experiences, as we expect they are more intuitive to understand than the probabilities that are actually stored in the risk evaluations. As the TuBE risk evaluation contains the information necessary to automatically convert the probabilities back to the unscaled numbers, this kind of implementation of the risk tolerance policies minimizes the number of transformations the reader must do themselves in order to follow the experiment. We note that the outcomes would not differ if the conversion were never made.

The policies are summarized in terms of experience counters in Table 6.2. Each formula applies to all assets. If the risk is not within the constraints for a single asset, the result of the evaluation is negative (deny access); we omit the gray area of delegated decisions in this experiment.

Table 6.2: The risk tolerance formulae.

Policy	Formula
A	maj pos + min pos − min neg − maj neg ≥ 0
B	maj pos + min pos − 3 * (min neg + maj neg) ≥ 0
C	3 * maj pos + min pos − min neg − 3 * maj neg ≥ 0
D	3 * maj pos + min pos − 3 * (min neg + 3 * maj neg) ≥ 0
E	maj pos − maj neg $\geq 0 \wedge$ min pos − min neg ≥ 0
F	maj pos − 3 * maj neg $\geq 0 \wedge$ min pos − 3 * min neg ≥ 0

As can be seen from Table 6.2, policies A-D (Basic, Pessimistic, Sharp and Sharp-pessimistic) are variations of the same formula, with the differences based on different multipliers for major and negative effects, while E and F (Separative and Separative-pessimistic) represent a different philosophy. The multiplier 3 is chosen to be sufficiently low for its effect to be balanced for a relatively short stream of experiences, while still being clearly different from the effect observed without a multiplier.

The left-hand sides of the inequalities presented provide us a one- or two-dimensional "score" that the trustee has within the decision policy, and which develops with their reputation. We will use these numbers to visualize the effects of the decision policy on the interpretation of the experience stream. For example, a trustee with 3 minor positive experiences and one minor negative experience would have a score of $3 - 1 = 2$ within policy A, but a score of $3 - 3 * 1 = 0$ within policy B. In both cases, the decision would be positive, but only barely so with the latter.

The policies A-D can be seen as using a form of utility measure, in the sense discussed in Section 4.2.1, that calculates whether the trustee has been sufficiently beneficial to the trustor in the past to trust them for the next action. In addition, E and F calculate two separate values, both of which follow a similar logic. All policies are friendly to newcomers, however, having a trust threshold set to 0 rather than a positive value.

The behaviour of the given policies on different experience streams is demonstrated as follows: A in Figure 6.1, B in Figure 6.2, C in Figure 6.3, D in Figure 6.4, E in Figures 6.5 and 6.6 and F in Figures 6.7 and 6.8.

6.2.2 Experience streams

We apply each of the decision policies defined in the previous section to a set of different experience streams to demonstrate their differences. This section defines the chosen experience streams, and the reasoning behind them.

As we are not focusing on the benefit of using multiple assets in this experiment, the experiences contain a known outcome only for the monetary asset, and unknown outcomes for the other assets. We will therefore refer to experiences directly by their monetary outcome; e.g. a "major negative experience" is an experience with a major negative effect on the monetary asset, and unknown effect on all other assets. Given that the tolerance formulae described above have their acceptance threshold at 0, all other assets will always be within the risk tolerance.

Each experience stream is produced by a generator based on a simple, deterministic repetition rule. For example, a principle of "every third experience has a major negative effect, the rest is minor positive" is expressed as a repetition rule of "2 x minor positive, major negative". In practice, exact cyclic repetition is rather unrealistic, but for the purposes of illustrating the behaviour of different policies via simulation, adding randomness would only produce unnecessary noise in the results.

We have chosen two simple experience streams as a baseline, one for a basic well-behaving actor (stream 1), who continuously completes minor transactions well, and one repeatedly misbehaving, or "sloppy" actor (stream 2), who never completes actions with a major effect but who cooperatively completes two minor actions for each major one. Besides representing an opportunistically misbehaving actor, the second stream can also represent a somewhat incompetent service provider. For example, the provider may offer very affordable cloud computing services, but the service does not scale up and hangs whenever it is fed a larger job. As a result, the service provider may be a good option for small tasks, but should never be trusted with larger ones.

In addition, we have calculated a pattern of optimal attacker behaviour against each of the six decision policies, and apply these streams to all of the policies as well (streams 3a to 3f). For policies C and E, we note that an attacker who is capable of cooperating on an action that causes a major positive effect for the trustee is also capable of building up its reputation faster. For this purpose, policies C and E have two optimal attacker patterns, one for a "strong" (i.e. more capable) attacker, and one for a "normal" (i.e. not as resourceful) attacker. The experience streams are summarized in Table 6.3.

Table 6.3: The experience streams. 1 and 2 are baseline streams, $3p$ are optimal attacker streams for decision policy p.

Stream	Repetition rule
Stream 1	All minor positive
Stream 2	2 x minor positive, (followed by a) major negative
Stream 3a	minor positive, minor negative
Stream 3b	3 x minor positive, major negative
Stream 3c, normal	3 x minor positive, major negative (= 3b)
Stream 3c, strong	major positive, major negative
Stream 3d	3 x minor positive, minor negative
Stream 3e, normal	minor positive, minor negative
Stream 3e, strong	major positive, major negative (= 3c strong)
Stream 3f	3 x minor positive, minor negative (= 3d)

We will return to the definitions of the optimal attackers below; for now, we present the experience streams as given. Each stream is 100 experiences long, which sets the length of each simulation round; the patterns defined are simple enough that the trends of the policy behaviour are clearly visible from 100 rounds of decisions.

As can be seen from the repetition rules, six of them form pairs of two equal streams. The experience stream 3c normal, representing the optimal behaviour of a weak attacker for policy C, is equal to the stream 3b; stream 3e for the strong attacker is equal to stream 3c for the strong attacker, and stream 3f is equal to stream 3d. This means that the optimal attackers in fact introduce only another 5 different streams to the experiment, and the total number of streams to apply to the decision policies becomes 7, containing the streams 1, 2, 3a, 3b, 3c strong, 3d and 3e.

6.2.3 Policy performance in the face of attacks

When it comes to selecting a trust decision policy to protect the trustor from harm, we can roughly divide the trustees into three categories:

- Clearly trustworthy actors, who mostly just complete actions with positive outcomes;
- untrustworthy or sloppy actors, whose service fails repeatedly for non-malicious reasons such as incompetence, misplaced cost savings or too high promises; and
- malicious actors, or attackers, who actively aim to defect as much as possible within the constraints of the decision policy used by the targets.

A good reputation system promotes the clearly trustworthy actors, weeds out the untrustworthy according to a suitable threshold, and is resistant to malicious actors, who in essence target the reputation system itself. If a malicious actor has full knowledge of its reputation and the trust decision policies of the target, it can optimize its behaviour so as to reap the most gains from defecting while keeping its reputation high enough that it is still allowed into further transactions. While locally stored reputation information is considered private and sensitive information, the reputation system design cannot *depend* on this information never leaking. Testing how well the system deals with an optimal attacker with full knowledge of the system gives us a measure of the performance of the system that does not depend on this unpredictable variable. This approach is used in evaluating e.g. cryptographical algorithms as well.

The two baseline experience streams (1, 2) of only positive experiences, and of positive experiences mixed with major negative experiences, correspond to the two first categories of actors. The malicious actors are represented by experience streams 3a to 3f, which represent the optimal way for the attacker to take advantage of its partners. As an attacker is similarly forced to weigh gains against benefits in deciding the targets for its attack, a decision algorithm minimizing the gains of the optimal attacker will also set a strict upper bound for the potential gains of any less optimal attackers.

This approach of aiming to minimize the maximal gains of an adversary is known as minimax decision strategy in game theory [RN03]. In security analysis, it is commonly used to evaluate defences, looking for ones that would minimize the maximal *damage* an attacker can cause under certain assumptions. There is a clear demand for it to be applied to reputation-based systems as well, when evaluating them from a security perspective [Gol11].

For the purpose of applying minimax decision analysis, we must define the rules of a game that assign costs and gains to different actions, or game moves, that the attacker can make, and aim to minimize its maximal gains. In our experiment, the modeled attacker has an option between either cooperating in an action with a minor or major effect, or defecting with a minor or major effect. His goal is to defect as much and with as high effect as possible, while keeping a reputation sufficient to prompt a positive trust decision also after the defection. Given our chosen decision policies, this means that before defecting, the attacker must first gain the reputation that the defection will cause him to lose.

In the defined game, we measure the attacker gains through the damage caused to the trustor, i.e. through the effect of its defections. By aiming to minimize these gains, we can measure how effectively a decision policy deters misbehaviour. From the point of view of security analysis, we argue that this is also a reasonable model for minimizing the maximal losses to the trustor due to the trustee's misbehaviour.

The reason we measure specialized attacker gains specifically, rather than trustor losses or even more general trustee gains, are twofold: First, in order to be able to analyze and influence attacker incentives, we exclude the losses the trustor suffers from turning down collaborations. It is already known that punishing misbehaviour is often detrimental to the individual on the short term, yet beneficial for the ecosystem and its members in the long term [FF03]. Second, as inter-enterprise collaboration is not a zero-sum game, we certainly do not wish to minimize the gains of everyone but the trustor; in fact, it would be best for the sustainability of the ecosystem to maximize the gains of all well-behaving actors instead.

To be optimal, the attacker takes advantage of full knowledge of the decision model used, and we will compare the decision policies based on how well the attacker is able to perform while being subjected to them. Actions with no reputation effect would be similar to doing nothing at all, which is uninteresting for the purposes of this demonstration; we therefore assume a compulsive attacker who must defect at least once per attack script.

We also assume that all outcomes are accurately detected, so the attacker's actions have a direct influence on its reputation. Also, as external reputation is not used here, fake positive experiences are not available as an option; we will return to this possibility in Section 6.3.

Whenever the attacker cooperates in order to increase its reputation, it must make an investment. For example, the eBay seller selling matches in order to defect on a car sale needs to buy and distribute the matches in order for the plot to succeed. As the attacker's aim is to defect, it benefits from successfully causing negative outcomes to the target, and must pay the investment to cause positive outcomes in order to increase its reputation. We scale the attacker's payoffs and costs of different outcomes as follows:

- A major negative effect: +6 points
- A minor negative effect: +2 point
- No effect: (not in use)
- A minor positive effect: -1 point
- A major positive effect: -3 points

The scaling between major and minor effects is chosen as a factor of three; this example weighing is sufficient to give a clear separation to the two different classes of outcome, and also matches the weight set earlier for major effects in the decision policies based on a utility measure. Negative effects, on the other hand, are considered worth relatively more than the corresponding positive effects cost, because it is assumed that the attacker does get some minor gains to balance the cost of cooperation as well, even though defection remains the central goal. We assume that the attacker is always working with a cooperative trustor, who does not defect itself as long as its trust decision is positive.

We also note that in addition to the weights here, causing 3 minor positive effects is assumed to be easier than causing a single major positive effect, because typically cooperation in major ventures requires more initial investment on the service capabilities of the attacker; for example in eBay auction terms, it is slightly easier to acquire many low-value items for sale than a few high-value items. This is not indicated by the scoring, but rather as a division between strong and weak attackers. On the other hand, causing a single major negative effect (i.e. defecting on a car sale) is speed-wise preferable over causing three minor negative effects (i.e. defecting on match sales), to ensure that the investment to good reputation is cashed in as profit as quickly as possible.

We will now present the optimal attacker scripts testing the strength of the decision policies. For each script, we show the series of actions that the attacker must take, and calculate the utility points gained by the attacker based on the scoring defined above. The more an attacker is able to gain, the weaker the policy is against knowledgeable malicious actors, as its deterring effect is not particularly high.

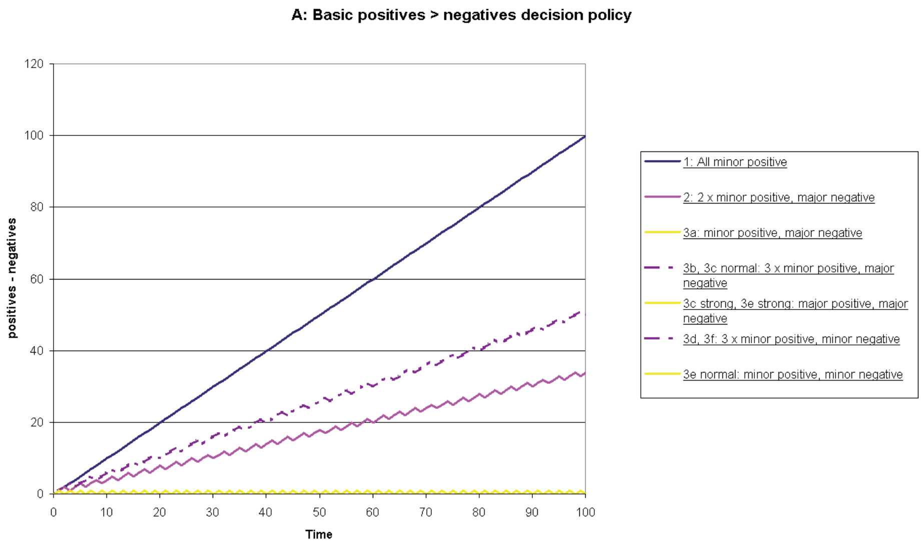

Figure 6.1: The behaviour of decision policy A. Streams 3a, 3c strong and 3e normal overlap, as do 3b and 3d.

Figure 6.1 presents the behaviour of decision policy A in response to the different experience streams. The optimal attacker script for decision policy A is as follows:

1. Cause minor positive effect
2. Cause major negative effect.

Given our scoring, the attacker ends up with $(-1)+6 = 5$ points gained per each iteration of the script, averaging 2.5 points gained per action taken. In other words, this is the maximal outcome the attacker can gain in our game.

The transaction rounds are set on the X axis, while the Y axis depicts the "trust score" gained by the actors representing the different experience streams, which in turn are depicted by the plots. The decision is always positive if the trust score is positive; the score is calculated as a side effect, and demonstrates how quickly each actor's reputation increases in relation to the chosen risk tolerance formula.

Policy A does not deny requests in any of the cases, as none of the streams have a larger number of negative experiences than positive. Some experience streams behave equally in relation to a given risk tolerance formula; Policy A makes the least separation between different streams, and divides the seven streams into four classes on the plot, all of which cause equally positive decisions.

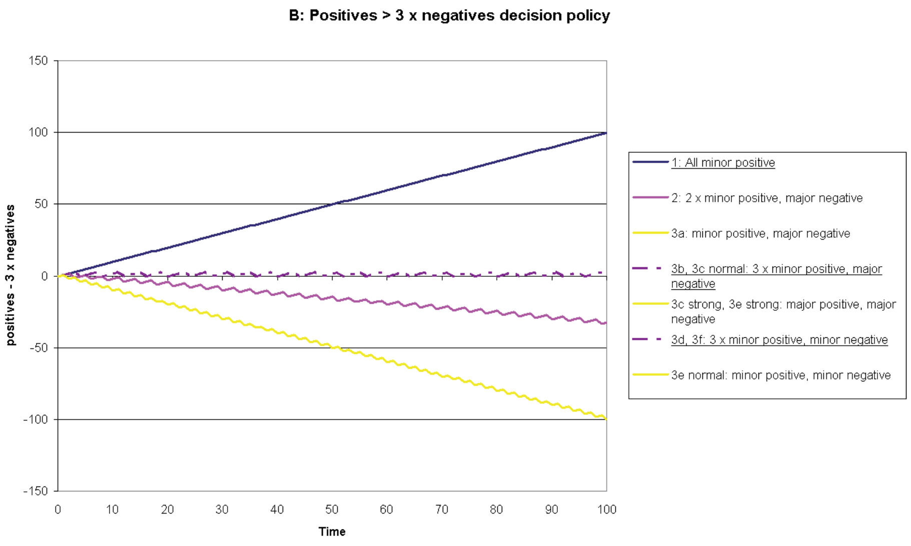

Figure 6.2: The behaviour of decision policy B. Streams 3a, 3c strong and 3e normal overlap, as do 3b and 3d.

Figure 6.2 presents the behaviour of decision policy B, given the different experience streams. The optimal attacker script for decision policy B is:

1. Cause minor positive effect x 3
2. Cause major negative effect.

With this policy, the relative weight of negative experiences to positive is higher in terms of reputation (3x) than in terms of the relative gain to the attacker (2x). The attacker's profits are therefore reduced, but he still gains $(-1) * 3 + 6 = 3$ points per each iteration of the script, averaging 0.75 points per action taken.

The streams with lines falling below 0 are rejected after the first round of the repetition rule; the lack of differentiation between major and minor effects is emphasized by the graph only showing four different lines for the seven streams; 3a, 3c strong and 3e normal are all seen as a single line, as well as 3b = 3c normal and 3d = 3f. Although B cannot differentiate between major and minor effects, it does reject the three cases (2, 3a, 3c strong = 3e strong, and 3e normal) where the number of positive experiences is not sufficiently larger than the number of negative. Two of these are also the two streams causing the highest losses to the trustor, while 3e normal (minor positive, minor negative) is, by our attacker gain definition, not quite as large a threat as the uncaught 3b = 3c normal (3 x minor positive, major negative) would be.

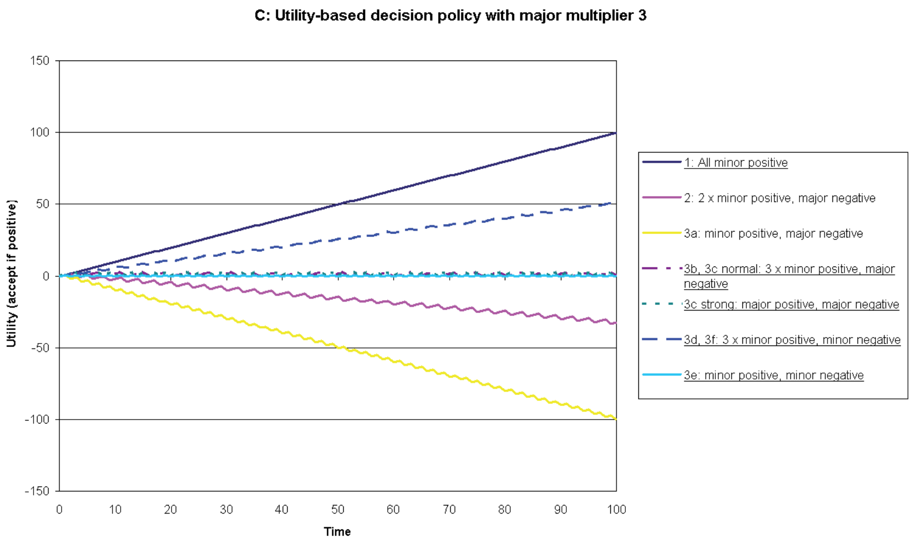

Figure 6.3: The behaviour of decision policy C. Streams 3b, 3c and 3e partially overlap.

Figure 6.3 shows the behaviour of decision policy C. The Y axis reflects the trust score of the actor, this time calculated based on the utility-based trust decision policy, with a multiplier 3 given to experiences with major impact.

The optimal attacker script for decision policy C is as follows:

1. Cause minor positive effect x 3 (/ cause major positive effect)

2. Cause major negative effect.

Similarly to B, the "normal" attacker ends up with $(-1) * 3 + 6 = 3$ points gained per each iteration of the script, averaging 0.75 points per action. Our preference for outcomes with a minor positive effect artificially reduces the speed of attacker gains in this case. A strong attacker script with a single major positive effect followed by a major negative effect would provide $-3 + 6 = 3$ points as well, but average 1.5 points per action taken.

Decision policy C successfully catches the two experience streams (2 and 3a) causing the highest losses. However, as it is less strict about negative experiences than policy B, it allows the stream 3e normal. It clearly improves on policy A, but still allows many attacker streams through.

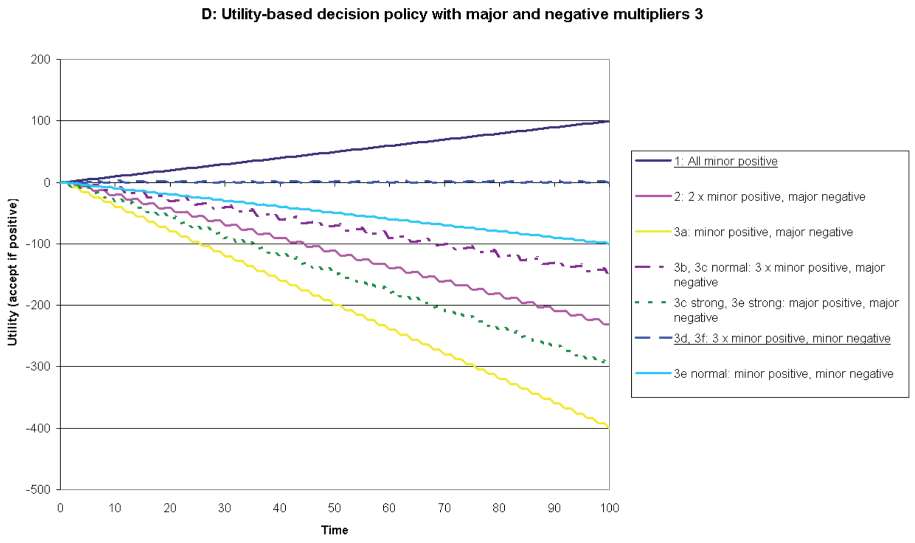

Figure 6.4: The behaviour of decision policy D.

The behaviour of decision policy D can be seen in Figure 6.4. The trust score on the Y axis is based on the utility-based trust decision policy, giving a multiplier of 3 both to major-impact and negative experiences.

The optimal attacker script for decision policy D is as follows:

1. Cause minor positive effect x 3

2. Cause minor negative effect.

With this policy, the relative weight of negative experiences to positive is less beneficial, as with decision policy B. When combined with a heavier reputation cost for major defecting, the attacker ends up making no profit at all. This means that the optimal script ends up being as close to no activity at all as possible, given that we did not allow the simulated attacker the option of full inactivity (0 points). The script with at least one defection leads to (-1) * 3 + 2 = -1 points, averaging -0.25 points per action taken.

Using 9 minor positive actions and a major negative would lose points faster than running the optimal script thrice, as it lasts only for 10 (vs. $4*3 = 12$) actions. The benefit per script would be the same: $(-1)*9+6 = -3$. With the single major defect, it averages to -0.3 points per action taken. A more powerful attacker causing 3 major positive effects for each major negative would lose points even faster: again $(-3)*3+6 = -3$ points per iteration, but averaging -0.75 points per action.

As decision policy D takes a particularly strict approach to defections, it successfully catches all experience streams that we have defined to be beneficial to the attacker. In addition, the policy also correctly orders the experience streams based on attacker preference: as we compare the streams' benefit to the attacker against the steepness of the downward slope, D orders the 5 rejected streams as 3a, 3c strong = 3e strong, 2, 3b = 3c normal and 3e. Stream 2 does provide more benefit per iteration (+4) than 3c strong = 3e strong (+3), but because the latter has a shorter script than the former, it allows the attacker to defect more often and gain a higher average benefit per round. The two streams that are not beneficial to the attacker are similarly presented in correct order.

Decision policy E divides the actions clearly into two groups, and therefore sets two options for the attacker: either he must invest in the capability to cooperate in major ventures in order to be able to defect with major impact, or he can avoid the investment but not be able to reap large gains fast either. The behaviour of the policy is presented in Figures 6.5 and 6.6. The graphs show the two different comparisons of minor and major effects, respectively. For a positive decision, a plot should be positive on both graphs.

Given our slight preference for minor positive effects, the optimal attacker script for decision policy E is as follows:

1. Cause minor positive effect (/ cause major positive effect)

2. Cause minor negative effect (/ cause major negative effect).

In this case, the attacker gains $(-1) + 2 = 1$ point per each iteration, averaging a gain of 0.5 points per action. If we assume a more powerful

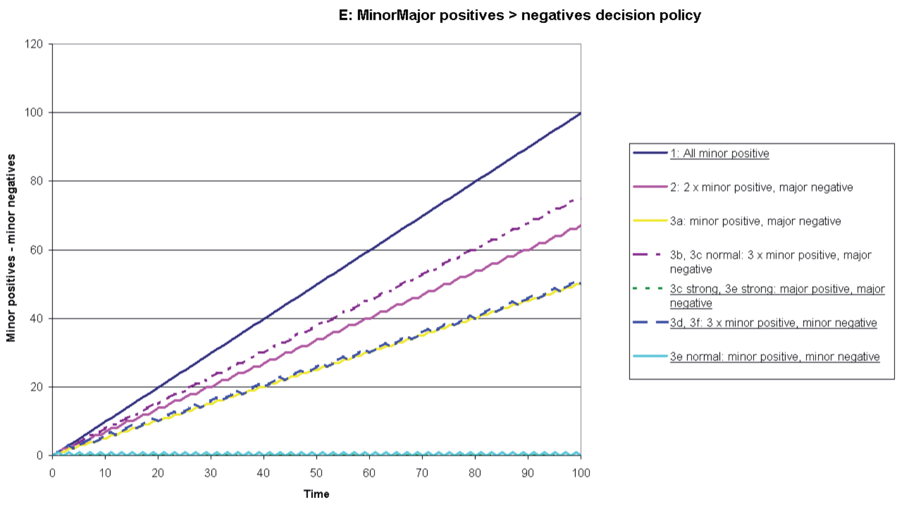

Figure 6.5: The behaviour of decision policy E for minor effects. Streams 3a and 3d partially overlap.

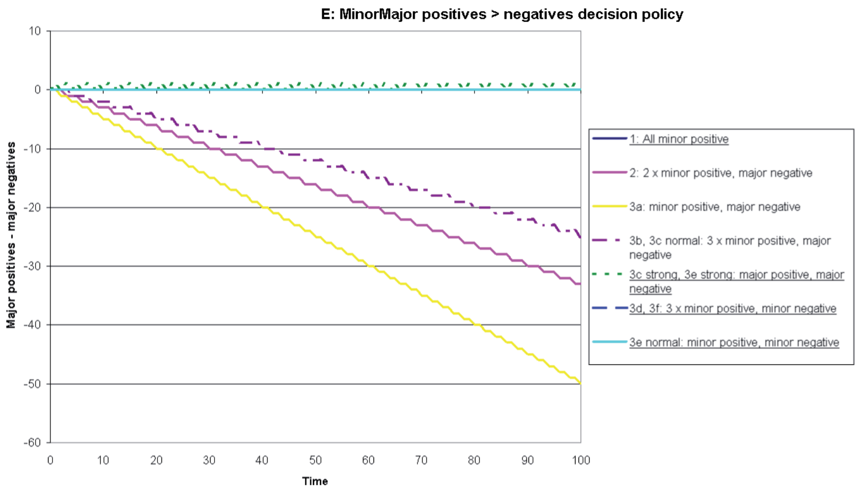

Figure 6.6: The behaviour of decision policy E for major effects. Streams 1, 3d and 3e normal overlap, and 3c strong partially overlaps them.

attacker and replace both minor effects with major ones, the gains increase to $(-3) + 6 = 3$ points per each iteration, averaging 1.5 points per action.

The minor effects graph shows that none of the experience streams have any problem upkeeping the balance of minor effects; as the attackers generally aim for large benefits fast, their defections show mostly on the graph plotting major effects. Like C, policy E rejects the two most attacker-friendly streams of 3a and 2, in addition to which it rejects 3b = 3c normal, which aims to exchange multiple minor cooperations for major defections. On the other hand, it happily accepts the slightly more lucrative tactic of alternating major positive and major negative effects, which is also the optimal strong attacker tactic against the policy (3c strong = 3e strong).

The behaviour of decision policy F is shown in Figures 6.7 and 6.8. The graphs split the presentation in two parts like the previous pair. The first graph shows the balance of minor effect outcomes, while the second graph shows the balance between the major effect outcomes.

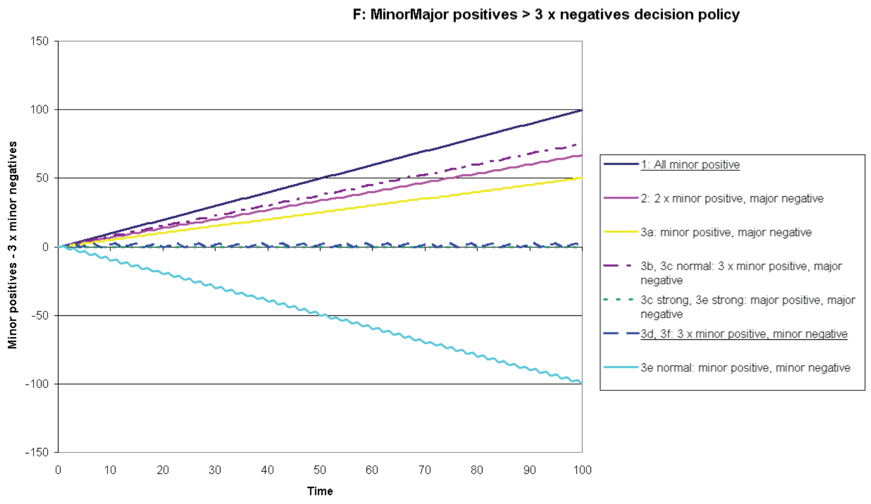

Figure 6.7: The behaviour of decision policy F for minor effects. Streams 3c strong and 3d partially overlap.

Decision policy F applies the same adjustment to E as policy D did to policy C. Actions with the smallest total effect become optimal once more:

1. Cause minor positive effect x 3
2. Cause minor negative effect.

As with policy D, the attacker is losing points as well: $(-1) * 3 + 2 = -1$ point per each iteration, averaging -0.25 points per action taken. Policy F differs from D mainly in that the option of trading minor positive actions for a single major negative is not available, although it was not the most attractive one to the attacker in any case, as it speeded up the losses.

Like policy D, F rejects all experience streams that would be beneficial to our defined attacker, accepting only the baseline good behaviour

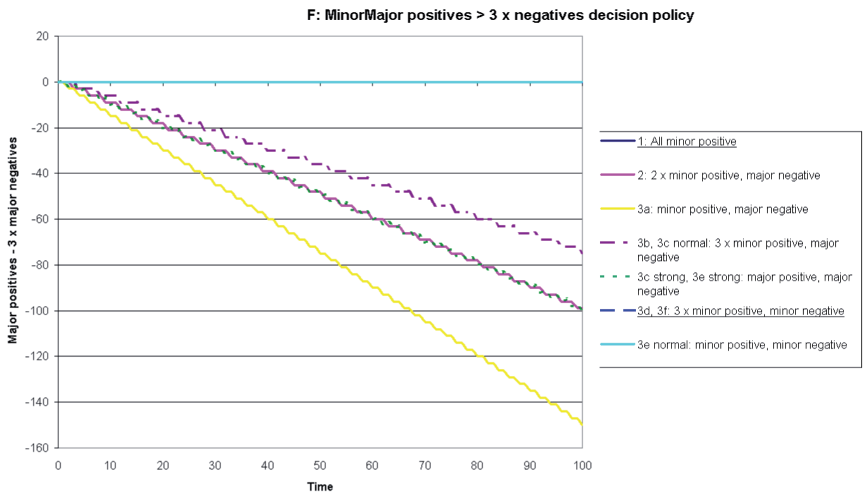

Figure 6.8: The behaviour of decision policy F for major effects. Streams 1 and 3e normal overlap, while 2 and 3c strong partially overlap.

stream (1) and the optimal but still lossy attacker stream (3d = 3f). In other words, it accepts only streams that do not form actual successful attacks. Due to the semantic gap between experiences with minor and major effects, policy F does not provide a clear ordering between the streams.

The curves for 2 (2 x minor positive, major negative) and 3c strong = 3e strong (major positive, major negative) partially overlap on the major half of the graph (Figure 6.8). The path of the former can be described as “two steps still, third step three down”, while the latter reads as “one step up, second step three down”; both eventually average out as a slope of -1 per step, synchronizing at every two (4 * 0 + 2 * (-3) = -6 in 6 steps) and three (3 * 1 + 3 * (-3) = -6 in 6 steps) iterations respectively.

6.2.4 Analysis

The six decision policies presented represent three approaches to differentiating between minor and major effects. The optimal attacker behaviour is an effective method at comparing the policies’ robustness; the scoring of the optimal attacker scripts is summarized in Table 6.4.

Awareness of the scale of the effect, i.e. the impact, in the policy pair C (Sharp) and D (Sharp-pessimistic) has attackers at a clear disadvantage when compared to A (Basic) and B (Pessimistic); the gains from malicious behaviour drop respectively. The strict scale division applied by the pair E (Separative) and F (Separative-pessimistic) further limits the attacker’s

Table 6.4: Optimal attacker performance with different decision policies.

Decision policy	Total points	Script length	Points per action
Policy A	5	2 actions	2.5
Policy B	3	4 actions	0.75
Policy C	3	4 actions	0.75 / 1.5*
Policy D	−1	**4 actions**	−0.25
Policy E	1 / 3*	2 actions	0.5 / 1.5*
Policy F	−1	**4 actions**	−0.25

*) The higher gains are possible for a powerful attacker capable of completing actions with a major positive effect.

gains in the case with equal weight given to positive and negative actions, but D and F already perform equally well against our attacker. On the other hand, the principle behind E and F is quite strict: if we consider actions causing major effects to be entirely in a class of their own, we are also unable to take advantage of earlier experiences gained with actions with only minor effects.

As can be seen from decision policies B, D and F, making the reputation cost of defection relatively higher than the monetary gains from it eventually defeats the purpose of malicious behaviour. This prospect relies on the assumption that all experiences require the actors to perform real actions, and all outcomes are correctly detected.

Despite the encouraging results of these specific simulations, relying only on fully reliable local experiences leaves a problem with newcomers: if the initial cooperation threshold for reputation is 0, like in our experiment, each newcomer can potentially defect for a single major negative effect with each cooperative actor in the environment before it is blocked from further collaboration with them. If the cost of creating a new identity is less than the gain from the round of defections available, malicious behaviour remains worthwhile.

The problem is slightly alleviated if a new actor is blocked from major ventures until it has gained enough reputation to cover a single defection, similarly to the attacker behaviour principle used in this experiment. This kind of adjustment is supported by the structure of the TuBE trust model, which makes it possible to use a different decision policy for different types of actions. If the risk tolerance policy were merged with the reputation update policy, i.e. storing reputation as a single number already fully preprocessed for decisions, it would not be possible to apply different decision policies on a single mass of gathered reputation.

No matter how strictly the decision policies react to defection or guard important actions, however, allowing the attacker even the first small action will inevitably allow him to defect in that one. In other words, at least a first minor defection will always be possible. Without this tradeoff, the network of actors would not be able to grow by introducing newcomers, because they would not start with sufficient reputation, and would not be able to gain a better one due to no one agreeing to collaborate with them. To allow newcomers, some risk of defection must be endured. The gains from these initial defections increase with the number of available targets.

The main way to reduce the gains from defections against different actors is communicated experience: in the optimal case, the attacker can only manage a single defection before word of its negative reputation spreads through the network. This reduces the payoff for creating new malicious identities considerably.

However, with communicated experiences, another type of misbehaviour becomes possible: actors in the network can lie about their experiences for various reasons. In order to control reputation-related malicious behaviour, a second type of misbehaviour analysis is needed to discourage lying and upkeep the quality of information provided [RKK07]. In TuBE, this type of knowledge is included as a credibility score for the information.

6.3 Experiment 2: Comparison of reputation update policies

Our second experiment compares the effects that four reputation update policies have on trust decisions. The policies operate on external reputation information, and the central choice to make is whether to accept information which is not entirely credible, and if so, what weight to give it in calculating the resulting reputation. The second experiment follows a structure similar to the first one: the policies under scrutiny are applied to a set of different simulated experience streams, some of which are optimized against each policy for the attacker to defect as efficiently as possible.

The reputation update policies have been selected to represent different types of solutions to this choice, and we again observe how effectively they discriminate against ill-behaved actors. The central choice and tradeoff in the first experiment concerned the number of possible partners; in this second experiment, accepting only high-credibility information may mean that we sometimes do not have any information available. This problem and the influence of the different policies is further analyzed in Section 6.3.4.

6.3.1 Reputation update policies

The policies for incorporating new experience aim to ensure a high quality of reputation information stored in the system. High-credibility, up-to-date information from well-trusted sources is what all trust decisions would be based on in a perfect world, but sometimes low-credibility information must be used to have some kind of baseline to work with. This opens the door for possibly counterfeited or otherwise useless reputation information. On a larger scale, low-credibility reputation networks can be subverted by malicious actors altogether, which means that they can falsely create a high reputation for themselves within that network. An example of this kind of reputation attack involves creating multiple accounts on eBay to pass positive feedback between them on transactions that never took place. This kind of Sybil attack [Dou02] can also be used to defame a honest actor with a burst of unfairly negative feedback from seemingly different sources.

A reputation update policy determines how to incorporate a new experience with a given, locally assigned credibility into the existing storage of reputation information. More specifically, in the case of external reputation, it determines whether the experience is incorporated into the current external reputation view, and what the new credibility value of the view is after the update. These two decisions are divided into two separate policies.

In our simulation, we do not use the final credibility value of the view in the trust decision; this makes the policy used for the view's credibility update only interesting when the value influences later reputation updates. Further, as only external experience is used, we do not define a separate policy for giving more or less weight to local experience; the resulting external experience views are converted to a risk evaluation as they are, independent of the view's final credibility value. For this reason, only one policy, which is using the view credibility as a parameter to the reputation update policy, actually depends on the choice of credibility update policy.

To control the scope of this experiment, we again define that the actor, action, its importance and decision context have no impact on the final trust decision. Unlike the first experiment, we focus on external experiences, which adds the credibility of each experience as input to the system. As earlier chapters have noted, local experiences all have the same credibility value of 1 (full credibility), while external experiences have varying credibility, represented with a value between 0 and 1.

As a further simplification, we use only two levels of credibility in the experiment: full credibility for information coming from "trusted sources", representing truthful experiences too expensive to counterfeit, and "low" credibility for information coming from sources which may have been compromised, representing counterfeitable experiences. We fix the trust decision policy to be the well-performing policy D from the first experiment, i.e. we require that for a positive trust decision, the number of experiences of different types satisfy 3 * major positive + minor positive − 3 * (minor negative + 3 * major negative) ≥ 0 for all assets. The reputation update policies will then determine which experiences are counted in these decisions.

The trust management system is given a stream of external experiences with given values, and after each experience has been processed by the reputation update policy, we transform the current reputation to risk and compare it to the risk tolerance formula described above to see whether the trust decision would be positive or not, and what the value of the left side of the above inequality, the "score" of the trustee, would be.

Each of the four reputation update policies are applied to the seven different experience streams. At the start of each stream, the reputation of the actor is purged to start the experience gathering from scratch.

There are two competing concerns related to selecting a reputation update policy: We simultaneously want to maximize both the amount and quality of reputation information available for making the trust decision. In other words, while we would like to accept only certified high-quality information to ensure well-informed decisions, we also want to ensure that there is always sufficient and up-to-date information available to make the decision. Sometimes a partner may only be well-known in a low-credibility reputation network, and the first whistleblowers warning that an actor has turned malicious may not generally be the most trusted sources.

The compared four reputation update policies are as follows:

I: Accepting: A basic reputation update policy that incorporates all experiences, independent of their credibility. This policy provides a baseline for comparisons; it embraces the first concern of maximizing available information, while ignoring the second.

J: Weighted: A policy transforming credibility directly to the weight the experience has on the view: for example, four experiences of credibility 0.25 would have the same effect as one experience of credibility 1. This policy follows the philosophy of accepting all available information, but ensuring that high-credibility information has the highest impact. It fulfils the first concern, while compromising on the second concern on ensuring the high quality of information.

K: Fixed-cutoff: A fixed minimum credibility requirement: if the credibility of the experience is below a fixed limit, it will be ignored. This policy focuses on fulfilling the second concern of maximal quality, and ignores the first.

L: Variable-cutoff: A variable minimum credibility requirement: if the credibility of the experience is below the current credibility of the reputation view, it will be ignored. The view's credibility is averaged over the experiences stored in it. This policy draws a compromise of the first and second concerns: after first ensuring that there is at least some information available, it focuses on ensuring maximal quality.

The *Accepting* policy I accepts all experiences and stores them in the reputation view. A suitable update policy for the view's credibility would be for example the average credibility of the experiences stored in the view. It represents the principle of gathering all information there is available on a given actor, regardless of its presumed quality.

While experiences are consolidated into the view by incrementing outcome counters and the separate credibility values are lost in the transformation, the new average credibility can be approximated with $(\text{Credibility}_{\text{view}} * n + \text{Credibility}_{\text{exp}})/(n + 1)$, where n is the number of experiences stored in the view. As said, this will not influence the simulation results, as view credibility is not used in trust decisions within the experiment.

The *Weighted* policy J implements an intuitive principle of weighing information by its credibility: more credible information should have more influence on updating the reputation view, so information which is not fully credible is weighed down by its credibility before it is added into the whole.

There is a slight technical issue with fitting this policy into our reputation model: the approach suggests that reputation views should store real numbers, while the reputation view counters are discrete by design. As an experience of credibility 0.5 is worth an 0.5 increase in the outcome counters, the policy object itself is used to store the "partial" experiences, and only whole units are incorporated into the reputation view. The temporary real values produced by this policy are kept invisible to the rest of the system, which operates on discrete experiences. The downside is that the policy object itself becomes stateful as a storage for temporary information.

Given an experience with a credibility of 0.5 and a major positive monetary outcome, the reputation update policy leaves the reputation view unchanged, and instead stores 0.5 in its own internal counter for major positive outcomes in the monetary asset. If the outcomes for other assets are unknown, the unknown effect counters are set to 0.5 for those. When a second similar experience of for example credibility 0.7 arrives, the internal counter for major positive monetary outcome rises to $0.5 + 0.7 = 1.2$, at which point 1 is deducted from the counter, and a major positive monetary outcome is added into a new experience with the same actor and stored into the reputation view.

The check is done simultaneously to all assets; if the incremented counter for an asset does not go over 1 to reflect a full experience, the outcome for that asset in the experience will be set to unknown. This also means that the number of unknown effects stored in the view may well become relatively more numerous than they were within the original experiences. As noted in earlier sections, experiences with unknown outcomes for specific assets can be used as a measure of the relative sparsity of information available for that asset, but they are not directly used in producing the risk estimate. The absolute number of unknown outcomes forwarded by this policy will not rise above the rounded-down sum of the credibilities of all experiences stored. If the information very often has unknown outcomes for certain assets while varying strongly within other assets, the counters for unknown effect can be disabled to stop the policy from forwarding experiences consisting only of unknown effects.

A suitable update policy for the view's credibility to complement policy J would be for example one which keeps the view credibility at a constant, predefined value. Averaging the credibility of the experiences stored loses some of its meaning when some of the incoming experiences only cause delayed changes in the reputation view.

The *Fixed-cutoff* policy K ignores experiences if their credibility is below a fixed level. For the purposes of this experiment, the cutoff point is 0.6, which means that the low-credibility experiences are ignored. It is a quite strong interpretation of the need to maximize the quality of decision-making information, as actors only known through low-credibility reputation networks will remain as completely unknown actors with this policy. On the other hand, the more questionable information from these networks cannot affect the trust decisions either. The choice may be justified for example in situations where some information sources are suspected to have been compromised.

As discussed before, the credibility value of an external experience is normally a combination of the credibility of the source network and the credibility assigned by the representative agent within the network, rather than fixed by the source alone. If the network supports credibility evaluation internally, this policy can also allow us to only accept only relatively high-quality information from a reputation network that occasionally produces very low-credibility information as well. A suitable credibility update policy to complement this kind of policy would be for example the averaging one described for the *Accepting* policy (I).

The *Variable-cutoff* policy L also ignores experiences with insufficient credibility, but its cutoff point varies based on the credibility of the information already stored in the reputation view. For the experiment, the policy is combined with the average credibility policy described for the *Accepting* policy (I): the view's new credibility is calculated to be $(\text{Credibility}_{\text{view}} * n + \text{Credibility}_{\text{exp}})/(n + 1)$.

This also means that if only low-credibility information is stored in the view, it will continue to accept more of it, ensuring that all actors can be covered. However, once it has access to higher-credibility information, it no longer accepts more of the low-credibility experiences. This allows for increasing the quality of the information over time once the set minimum has been reached. As a downside, it will not react at all to low-credibility warnings about behaviour changes once some high-credibility information has been stored as well.

The behaviour of the given policies on different experience streams is demonstrated as follows: I in Figure 6.9, J in Figure 6.10, K in Figure 6.11, and L in Figure 6.12.

6.3.2 Experience streams

We apply each of the reputation update policies described above to a set of different experience streams to demonstrate their differences. Each stream is produced by a generator based on a deterministic repetition rule, such as with the experiment described in Section 6.2.

We will continue to focus on the monetary asset. A "major positive experience", for example, refers to an experience with a major positive outcome in the monetary asset, and unknown outcomes in all others. The unknowns will have no side effects on the behaviour of the policies used, as they are ignored by the decision policy. It is worth noting, however, that as a result of the structure of the experiences, the *Weighted* policy (J) will merge some experiences with only unknown outcomes into the view.

As mentioned earlier, the experiences are all external, with two levels of credibility: full (1) and low (0.5) credibility. For brevity, we also refer to these two types as "real" and "fake" — this is not to indicate that the credibility value clearly denotes validity, but due to the nature of the experiment: the low-credibility experiences in most cases are chosen to represent counterfeit experiences produced by attackers. The system cannot tell apart fake and real experiences: it only sees differences in assigned credibility.

The full credibility experiences could also be fed into the system as local experiences, but then they would not go through the observed update policies, as they are only applied to external experiences. The attacker's defections are therefore not actually directed at the simulation's trustor itself; the defections can instead be seen as reported attacks against the "trusted" neighbours of the trustor, whose shared experiences are considered fully credible. The low-credibility experiences, on the other hand, would come from a more open network, which is more likely to have compromised actors producing incorrect information besides any factual reports.

We have chosen three streams as a baseline for demonstrating the general behaviour of the policies: a well-behaved actor known through a reliable source, a seemingly well-behaved actor known only through an unreliable source, and a well-behaved actor suffering from a reputation attack defaming it in an unreliable reputation network. An additional four streams represent the optimal attacker behaviour against the four different policies, bringing the total number of streams to 7. The optimal attacker behaviour is further discussed below; we present the streams here as given. Each stream is 100 experiences long; the trends of the behaviour of the policies can be observed from the corresponding 100 rounds of trust decisions. The experience streams are given in Table 6.5.

Table 6.5: The experience streams. 1-3 are baseline streams, $4p$ are optimal attacker streams for reputation update policy p.

Stream	Repetition rule
Stream 1	All real* minor positive (credibility 1)
Stream 2	All fake* minor positive (credibility 0.5)
Stream 3	Reputation attack: real minor positive, fake minor negative
Stream 4i	3 x fake major positive, real major negative
Stream 4j	6 x fake major positive, real major negative
Stream 4k	3 x real minor positive, real minor negative
Stream 4l	$3r$ x fake major positive, r x real major negative**

*) Experiences with credibility 1 are referred to as "real", while experiences with credibility 0.5 are referred to as "fake" for brevity.
**) Here r indicates the total number of defections; the script runs only once.

Stream 1 represents information about a well-behaved trustee who has interacted with a reliable source. In order to be useful, a reputation update policy must interpret this information as a sign of trustworthiness. Stream 2 can represent a well-behaved trustee who has only interacted with sources that are not inherently trusted, but also an attacker who has control over a less reliable information source and can therefore produce counterfeit positive experiences on itself; the information will also be treated differently by the different policies.

Stream 3 represents a reputation attack against a trustee who behaves well in its interactions with reliable sources, but is constantly defamed by unreliable sources. It would be desirable that a reputation update policy would put more weight on the positive information from reliable sources. On the other hand, it is also entirely possible that an actor behaves differently towards different actors; if two sources are in disagreement, the usefulness of the information is altogether slightly reduced — either the trustor is an unpredictable turncoat, or at least one of the sources is lying. There is no separate stream for demonstrating a reputation attack made by a high-credibility source, but the effect can be imagined through combining this stream with the *Accepting* policy (I).

The attacker streams are similar to the previous experiment's attacker streams; the main difference is the introduction of credibility information. In addition, stream 4l is not really a repetition rule, but runs only once: for 100 rounds, r is set to 25, and the script becomes 75 x fake major positive, followed by 25 x fake major negative.

In the attacker scripts, all low-credibility experiences are seen as counterfeit, while the high-credibility experiences are considered to truthfully describe the outcomes of actions. We wish to promote a clear preference for high-credibility sources when such information is available, while being aware of the tradeoffs. More complex policies should put weight on how credibility values are assigned in the first place, but share the preference for high-credibility information in the actual updates.

6.3.3 Policy performance in the face of attacks

When selecting a reputation update policy to protect the trustor from being mislead by external reputation information, we can roughly divide the trustees into four categories:

- Well-reputed actors recommended as trustworthy by high-credibility sources,
- Promising actors recommended as trustworthy by low-credibility sources, but generally unknown by high-credibility sources,
- Shunned actors warned to be untrustworthy either by high-credibility sources or by unanimous low-credibility sources, and
- Mysterious actors receiving either very few or contradictory recommendations.

While all of these categories are simply perceptions rather than watertight proof of the trustees' actual behaviour and trustworthiness, a good reputation system should generally promote the well-reputed actors and weed out the shunned actors. However, the two other classes require more careful balancing. A very risk-averse trustor will prefer not to collaborate with the mysterious actors, independent of whether they offer better terms of service. Should everyone adopt this approach, though, newcomers will have no chance of proving themselves, targets of defamation cannot clear their name, and the service ecosystem will begin to deteriorate. The promising actors face a problem similar to newcomers in that they have not proven themselves enough, but at least they have some recommendations supporting them. On the other hand, it is also easier for a malicious attacker to appear as one of the promising actors rather than a well-reputed one, and even easier to claim that any negative recommendations about it result from reputation attacks rather than honest feedback.

The first experience stream represents the well-reputed actor category, while the second stream represents promising actors. We have skipped the shunned actors category in the simulations, as all policies would leave them out[3], and have set the third stream to represent one type of mysterious actor, one receiving contradictory recommendations. Additional streams represent optimal attackers targeting specific reputation update policies.

As with the first experiment and the minimax analysis, the optimal attacker has full knowledge of the policies in use, and aims to keep their reputation high enough to ensure a positive trust decision. In addition to the options before, the attacker now has control of a malicious recommender, a reputation source which produces external experiences to the trustor. The source has lower credibility than the trustor's first-hand experience; for the sake of the example, we set the credibility of these counterfeit experiences to be 0.5 in the trustor's system, which is half of the credibility granted to local experiences and external experiences from trusted sources (1).

We define counterfeiting these low-credibility experiences to cost nothing to the attacker; once it has gained control of third party recommenders, it simply has them conjure as many positive experiences as it needs. On the other hand, having to produce more information does slow down the attacker. We provide scoring for the policies based on how many actions, counterfeiting or causing actual experiences, it takes in total to produce the gain for the iteration, which is reduced by having to counterfeit more experiences.

[3]To the trust management system, credible negative feedback would appear as locally observed misbehaviour, which was simulated in the previous section.

We have also examined a weak attacker variant which is only able to produce counterfeit experiences and must defect from all real actions. However, barring the *Fixed-cutoff* policy (K), the weakness makes no actual difference, and with K, it cannot gain any reputation.

We will now present the optimal attacker scripts testing the strength of the reputation update policies. For each script, we show the series of actions that the attacker must take, and calculate the utility points gained by the attacker. Again, a measure of the deterrence strength of the policy is how much and how fast the attacker is able to gain by defecting.

The first update policy sees no difference between low-credibility and high-credibility information. Figure 6.9 presents the behaviour of policy I when applied to the different experience streams. The X axis represents the rounds of transactions, and the Y axis represents the trust decision utility measure or "trust score", calculated based on the *Sharp-pessimistic* trust decision policy D; as noted, the trust decision is positive while the score remains above 0.

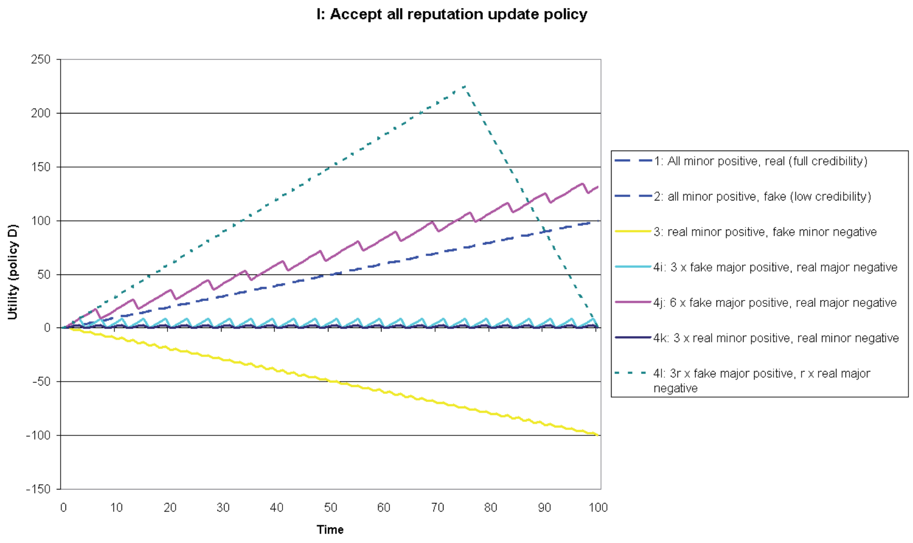

Figure 6.9: The behaviour of reputation update policy I. Streams 1 and 2 overlap, while 4i and 4k partially overlap.

In this graph and the following, the X-axis again depicts the number of rounds, and the Y-axis the trust score calculated according to the utility-based trust decision policy D defined in Section 6.2.1.

The optimal attacker behaviour for the *Accepting* reputation update policy I becomes:

1. Counterfeit major positive effect x 3
2. Cause major negative effect.

Given our earlier scoring, the attacker gains $3 * 0 + 6 = 6$ points per iteration, averaging 1.5 points per action (6 points per authentic action and 2 points per counterfeit action).

As streams 1 and 2 differ only by credibility value, policy I is unable to differentiate between them; they appear as a single line in the figure. The policy is also unable to identify any of the attacker behaviours 4i, 4j and 4l as attacks, while falsely identifying stream 3, which represents a reputation attack against the trustee, as the trustee's misbehaviour. Stream 4k is an attempted "attack", but according to our scoring model it is actually not beneficial to the attacker; it causes positive decisions as expected.

The *Weighted* update policy slows down the attacker slightly by scaling down the impact of low-credibility experiences while still accepting all information; more counterfeit experiences are needed to upkeep a good reputation. Figure 6.10 shows the behaviour of policy J when applying the different experience streams.

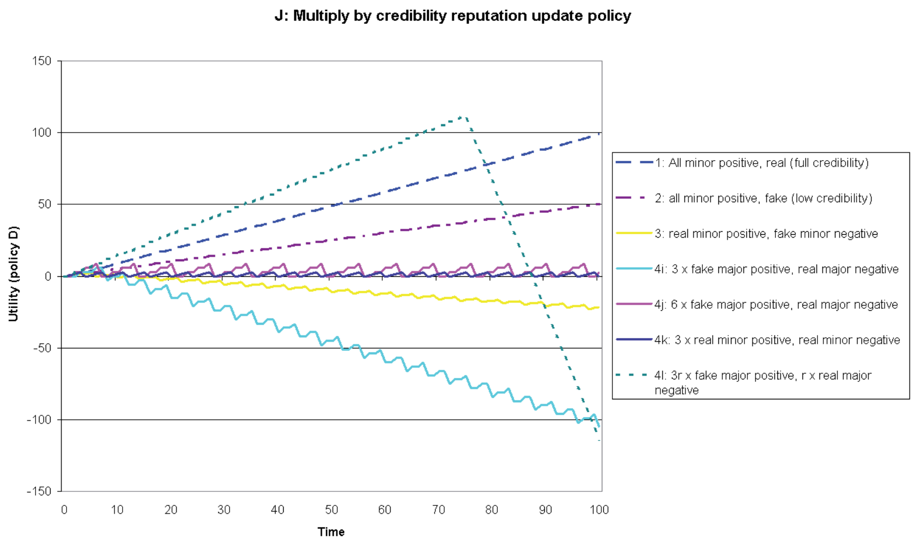

Figure 6.10: The behaviour of reputation update policy J.

The optimal attacker behaviour for reputation update policy J is therefore:

1. Counterfeit major positive effect x 6
2. Cause major negative effect.

The attacker still gains 6 * 0 + 6 = 6 points per iteration, which now averages to 0.85 points per action (6 points per authentic action and 1 point per counterfeit experience). In the general case, if the credibility of the counterfeit information is $0 \leq c \leq 1$, the attacker can gain $2 * c$ points per counterfeit experience produced; the higher-credibility experiences it can counterfeit, the greater its gains. This emphasizes the need to assign credibility values wisely as a part of defending against malicious actors.

Given the credibility value of 0.5, stream 2 increases the trustee's reputation by 1 every second action; as a result, the number of positive stored experiences is half of what stream 1 causes.

The reputation attack represented by stream 3 remains successful: while fake experiences only count as half experiences, two fake experiences together have a weight of 3 (0.5 * 2 * 3) in the decision policy, while two positive real experiences only have a weight of 1 * 2 * 1 = 2. While the different weights affect its success here, in principle a reputation attack like this can only be slowed down by a weighted policy like J.

The simple attack of stream 4i is detected, as there are too few counterfeit positive experiences in relation to the actual negative ones. Doubling their number for 4j to balance for the lower weight generates the optimal attack against this policy. Meanwhile, earlier reputation gains are already spent halfway through the defection phase of stream 4l.

With the *Fixed-cutoff* reputation update policy K, the credibility limit for accepting an experience is set to be higher than the credibility of the malicious source; in the opposite case, the results would be the same as with policy I. The behaviour of reputation update policy K is shown in Figure 6.11.

Because of the fixed cutoff, the attacker can only increase his reputation by authentic actions, which reduces his optimal behaviour to the non-beneficial script seen in the first experiment:

1. Cause minor positive effect x 3
2. Cause minor negative effect.

As noted earlier, the attacker is now losing points: $(-1) * 3 + 2 = -1$ point per each iteration, averaging -0.25 points per authentic action taken; without counterfeiting this is not a successful attack.

As all low-credibility experiences are ignored, the policy gives no credit for them, successfully identifying all attacks. The reputation attack represented by stream 3 has no ill effect on the trustee's reputation, while the defections of streams 4i, 4j and 4l cause the trust decisions to promptly

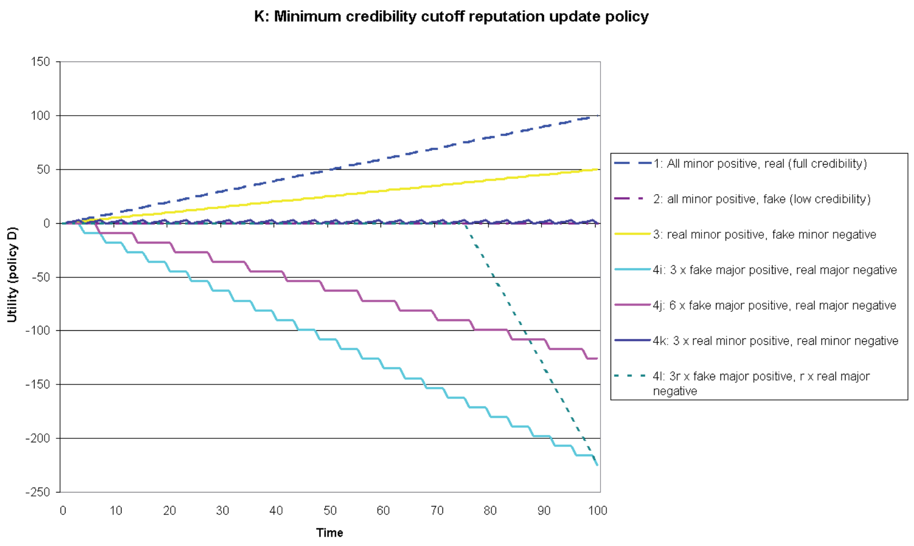

Figure 6.11: The behaviour of reputation update policy K. Streams 2, 4k and 4l partially overlap.

turn to negative. The optimal "attack" stream, 4k, oscillates near the 0 line, and is not beneficial to the attacker.

It is worth noting that if we interpret stream 2 as an actor who is genuinely cooperative, but only known through a low-credibility information source, this reputation update policy is unable to take advantage of the information: the trustee remains as good as unknown, gaining no reputation in the local system. This is also the main weakness of policy K: more strict selectivity means that there is less information available.

With the *Variable-cutoff* reputation update policy L, the attacker can only take advantage of counterfeit experience until the trustor has gained credible enough local information that it refuses to accept further low-credibility experiences. However, as the malicious source's credibility never goes down within the scope of the experiment, the policy does not actually limit the attacker's options much in practice; it only forces the attacker to make more long-term investments in its reputation. The behaviour of reputation update policy L is depicted in Figure 6.12.

With this policy, the attacker has no actual limit of how many counterfeit experiences it can pass to the trustor until it starts defecting. However, after the first defection, the trustor has its first experience above credibility 0.5, and this averaged with the earlier experiences will inevitably bring the cutoff point above 0.5. This only changes the ordering of actions in execut-

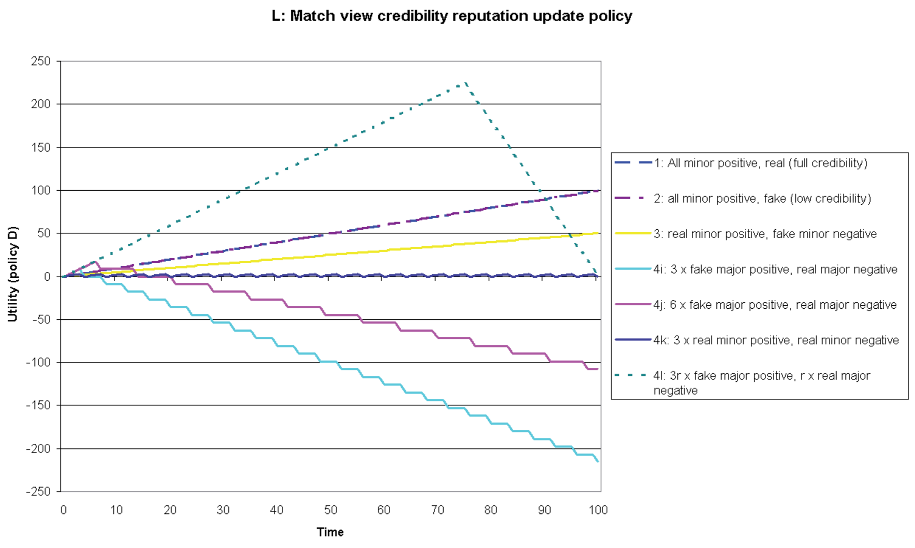

Figure 6.12: The behaviour of reputation update policy L. Streams 1 and 2 overlap.

ing the attack, however; the relative gains remain unchanged. The optimal attack becomes:

1. Counterfeit major positive effect x 3 (x r)

2. Cause major negative effect (x r).

As with the *Accepting* policy I, the attacker gains $3 * 0 + 6 = 6$ points per iteration, averaging 1.5 points per action (6 points per authentic action and 2 points per counterfeit experience). Once the initial feed of counterfeit experience has been spent, this policy would then force the attacker to follow the script described in policy K in order to be able to defect any more.

The main difference between the attacks against policy I and L is that for L, the attacker must decide beforehand how many rounds of the script it wishes to run, and feed the trustor all the counterfeit experiences for the run before it begins the defection half. Such a long-term investment does make the attack slightly less reliable, but here it the difference is not measurable. In fact, this attack demonstrates the need to react swiftly to behavioural changes: because the decision policy gives equal weight to the old and new experiences, the attacker can spend its gained reputation in peace. If a sudden behavioural change caused a reputation epoch change, the weight of old reputation would no longer allow a long defection phase. This approach is demonstrated in a separate publication [RHK11].

The reputation view's credibility is calculated as a weighted average of the earlier experiences and the new experience. As with policy I, no difference between streams 1 and 2 is apparent to the chosen trust decision policy. However, during and after stream 1, the reputation view's credibility is a solid 1, while with stream 2, it is only 0.5; this could be considered in the trust decision by a suitable policy.

The reputation attack of stream 3 fails from the beginning, as the first experience already sets the view's credibility to 1, cutting off anything less credible. Streams 4i and 4j gain positive reputation from their counterfeit experiences up until the first defection, which brings the average credibility of data stored in the view above 0.5. After this, the policy only accepts the actual experiences with credibility 1, finally ending up with average view credibilities of circa 0.95 and 0.85, respectively. The difference is caused by full-credibility experiences appearing more often in stream 4i.

The cutoff of counterfeit experiences after the first defection forces the optimal attacker (represented by 4l) to gain all of its positive reputation before it begins to defect. Finally, stream 4k, which is not actually beneficial to the attacker due to consisting only of real experiences with relatively few defections, keeps the trust score nonnegative and close to 0 as with the *Fixed-cutoff* policy K.

6.3.4 Analysis

The four reputation update policies presented each represent their own approach to differentiating between low-credibility and high-credibility information. We have compared their behaviour in relation to the different experience streams in the previous section. Table 6.6 compares the policies based on how well they thwart the optimal attacker.

The mean length of the scripts is equal to the mean length of the scripts in the first experiment, and the length of the attacker script for the *Sharp-pessimistic* decision policy D in particular. The attacker script for the *Weighted* policy J is clearly longer, while the one for the *Variable-cutoff* policy L shares the 4-step cycle in principle but is forced to delay its payoffs or it becomes unable to repeat the script more than once.

Typically, the attackers seem to have have more to gain from the extended information sources. The *Fixed-cutoff* policy K is the only one which performs as well with the external information sources as policy D originally performed with only local information. This is simply because K ignores the low-credibility information source altogether.

Table 6.6: Optimal attacker performance with different reputation update policies. Script length is expressed as the number of actions involved.

Update policy	Total points	Script length	Points per action
Policy I	6	$3 + 1 = 4$	1.5 (2 / fake, 6 / real)
Policy J	6	$6 + 1 = 7$	0.85* (1*, 6)
Policy K**	-1	$0 + 4 = 4$	-0.25 (0, -0.25)
Policy L***	$6r$	$3r + r = 4r$	1.5 (2, 6)

*) A lower value of “low” credibility reduces payoff speed.
**) Degrades to policy I if the minimal credibility cutoff point is too low.
***) The repeatability of the attack script is limited.

It is clear that using information that comes with the possibility of maliciously introduced errors is a tradeoff. As we argued in the analysis of the first experiment, information sharing and complementing local information with external is the only way of reducing the benefits of circulating attacks within a network of actors. On the other hand, uncritically accepting only negative information and ignoring the low-credibility positive in case it is counterfeited makes reputation attacks a viable method of shutting off competitors from the market. Granted, a reputation attack at a trustee is not a direct attack against the trustor, but it will leave them stuck with fewer service providers, or at worst none to choose from.

Out of the four policies, the *Accepting* policy I is most vulnerable to low-credibility counterfeit experiences: it ignores credibility information altogether. The policy is too accepting, and as such suits only environments where the vast majority of information is truthful, and the occasional error will not disrupt anything critical.

The *Weighted* policy J also accepts all information independent of its credibility, and as such is not far from the *Accepting* policy I. The weighing approach does have the effect of slowing down attacks based on counterfeiting. However, in environments where fake experiences are free and can be quickly mass-produced, this policy will not make much of a difference. This combined with the fact that the policy in practice needs to upkeep its own information makes it poorly suited to very large and active networks of actors. Where massive flows of shared experiences are normal, its moderate slow-down effect is easily overrun by determined attackers, and having to upkeep a handful of additional state information for each actor whose reputation is observed makes it more expensive than the options.

The *Fixed-cutoff* policy K very efficiently hampers even the optimal attacker, but as said, it also suffers from being too strict — and if it were any less strict, it would not make a difference. Similarly to the problems with the *Separative* decision policies E and F noted earlier, policy K makes an absolute separation between two different classes of information, and as a result is unable to take advantage of connections between the classes. On the other hand, if credibility values are well-assigned and the environment has a considerable number of malicious information sources, policy K is a natural choice for pruning the incoming experiences. In fact, in its division of experiences into groups based on credibility, K behaves similarly to reputation systems which actively select the recommenders they ask for information in the first place [RKK07].

Compared to policy K, the *Variable-cutoff* policy L is able to use information more efficiently in situations where it is most sorely needed: if only low-credibility information is available, L takes advantage of it until it gains higher-quality information. It fares relatively poorly against the optimized attacker, mainly because the simulated attacker can risk a long-term investment into its reputation. After the first credible defection, however, the policy has better information available and refuses to accept more low-credibility experiences. The optimal attack is vulnerable to interruptions; one interruption could come from successfully detecting behavioural changes and reacting to them promptly, at which point the sudden first defection would cause a more dramatic drop in the trust score. We discussed the division of reputation into epochs in Section 4.1.5.

The policies presented here are only a small subset of options, chosen both to represent different takes on the principles of maximizing the amount and/or quality of information, and for their suitability to an experiment using simple experience streams. A simple option is often the best, especially considering that the overall system is quite complex even with very simple partial policies. On the other hand, the reputation update policy is a central guard of external information flowing into the system, and a decision system is only as trustworthy as the data it builds on. With this in mind, we propose some extensions to the presented policies for specific purposes.

As argued above, the *Weighted* policy J does not perform very well by itself. It may be that the principle of weighing experiences based on their credibility would serve better if complemented by a policy with some actual selective power, such as the cutoff policies K or L. For example, K could be extended to reject experiences below a minimal credibility value c_1, unquestioningly accept experiences with credibility above another value c_2, and multiply experiences in the "gray area" between the two borders ($c_1 \leq c \leq c_2$) with their credibility or a fixed value in order to reduce their influence. Within this experiment, this kind of extension would not have had much effect due to the extremely limited variation between credibility values; it would simply have slowed down the attacks, as it already originally did.

If storage space is an issue, policy J could be approximated with a variant that does not store internal state, but would be probabilistic rather than deterministic in its behaviour. This variant policy accepts each experience with a probability given by the credibility value rather than adding up partial experiences. Such an algorithm behaves on the average more predictably as the number of experiences processed per actor grows. However, this increase does not immediately follow from there being more active actors in the network, so there are situations where storage space does become an issue but a probabilistic version of policy J could still be inconveniently unpredictable.

The instant increase in selectivity may in some situations be an issue with the *Variable-cutoff* policy L, which will basically refuse all experiences below the credibility of the first experience presented to it. A straightforward method to make the policy more flexible involves adjusting the barrier of acceptance downwards based on the number of experiences stored in the view; generally, once a certain amount of information has been reached, rejecting new low-credibility experiences seems less limiting. For example, up to a suitable minimum number of experiences, such as 20, the extended policy may qualify information that has only at least half the credibility of the view, or go as far as accepting all information, or all above a minimum credibility (as per policy K) until a desired number of stored experiences is reached, and then applying policy L from that point onwards to adjust to the quality of information available. In the general sense, the experience with credibility $\text{Credibility}_{\text{exp}}$ would be accepted if $\text{Credibility}_{\text{exp}} \geq f(n) * \text{Credibility}_{\text{view}}$.

In this case, the behaviour of the above kind of policy can be easily enough deduced from the behaviours of the existing simulations. We have explored some basic ideas for a multiplier function $f(n)$ that depends on the number of experiences stored. However, policies with simple "intuitive" explanations likely to be understandable by the users were not otherwise particularly compatible with the simulation setup. For example, it would seem reasonable to demand that the change of the view's credibility should satisfy $\text{Credibility}_{\text{view}} * n \leq \text{Credibility}_{\text{new}} * (n + 1)$, where n is the number of experiences stored in the view and $\text{Credibility}_{\text{new}}$ the new credibility of the view. It turns out to not be strict enough: if combined with the average credibility update policy we have used earlier, this policy would only ever reject experiences with a credibility value below 0, which means accepting everything.

Replacing the actual new credibility in the inequality with for example an unweighted average of $(\text{Credibility}_{\text{view}} + \text{Credibility}_{\text{exp}})/2$ would give $(n - 1)/(n + 1)$ as the multiplier function $f(n)$. This function behaves in practice rather similarly to the original policy L given our simulation setting, as an experience of credibility 0.5 will only be accepted into a view with a credibility of 1 up until 3 experiences have been stored. While this does not directly indicate that the multiplier function is without merit, in terms of understandability it would seem a better idea to set fixed points which adjust the minimum credibility requirement, if such an adjustment is needed.

In summary, changing the *Variable-cutoff* policy to adjust to the amount of experiences available seems to lead to mathematically more complex policies without providing much impact in terms of producing different results than the simulated, simpler policies.

The reputation epochs discussed in Section 4.1.5 are managed by a different policy than the ones simulated here: a reputation epoch change policy takes effect after the decision to include an experience has been made. While they have been left out of the simulations here for this reason, either one of the suggested epoch change policies, combined with an epoch-aware trust decision policy, would be sufficient to thwart the lengthy optimal attack against the *Variable-cutoff* policy L. While the general problem of oscillating behaviour is not entirely eliminated even with reputation epochs, it forces the cycles of attack versus reputation-gaining good behaviour to be short, like most simulated streams here. We have studied the effects of reputation epochs and suitable policies in a separate paper [RHK11].

Different options for reputation update policies suit different types of information sources and situations, and they must be fitted to the other policies in use. It would seem to be a quite serviceable approach to accept information from certain known sources nearly uncritically, and spending the effort analyzing borderline credible information if and only if the extra experiences are interesting enough given the information already available. In the end, having the view credibility information available at decision time leaves us with some additional flexibility. If the trust decisions for some actions require a highly critical view of the trustee's reputation, we can refuse a request if we do not have credible enough information available, while taking full advantage of the low-credibility information for less critical trust decisions.

6.4 Related work on simulation experiments

Our experiments make two kinds of contributions: First, the behaviour of a given decision or reputation update policy is demonstrated through exposing it to different representative experience streams and plotting the resulting trust decision score. Second, the limitations of each policy are demonstrated by defining the behaviour of an optimal attacker, and calculating how much it is possible for it to benefit by defecting before its reputation drops too low. We claim that for studying the attack resistance of policy-based systems such as TuBE, this approach is more suitable than the prevalent comparable methods applied in related work.

As the objective correctness of a reputation-based trust management system's calculations cannot be proven, given that there is no suitable definition for "correct" in this context, the results that the system produces cannot be validated directly. Both our test runs with experience streams and the scoring for attackers are forms of simulating the system on fabricated data; "real data" for this purpose does not exist due to the subjectivity of the system, and even remotely realistic data with a sufficient level of detail is unattainable at this time. While our example simulations weigh little in way of proof of correctness, we extend the state of the art in simulation experiments for reputation-based trust decisions particularly through our game theoretic minimax investigation of optimal attackers.

To discuss the state of the art in simulation experiments, we present experimentation approaches from two categories: simulating marketplace resistance against attackers following given behaviour patterns, and simulating a single actor's competitiveness in a marketplace.

The first category represents mechanism design: it compares the behaviour of different reputation systems in a marketplace where a number of agents misbehave, and the goodness of the system is based on how well the entire marketplace can resist different kinds of misbehaviour when everyone uses the same system and the same policies. This simulation has the involved actors using reputation information to select their transaction partners probabilistically, which adds an element of randomness into the simulation. As the defined attackers follow statistical models rather than making optimal choices according to the decision context, this kind of simulation is directed more towards the domain of reliability than security [Gol11].

The second category represents agent design; it pits different decision policies against each other on the same marketplace. It forms a type of benchmark for how a given decision model influences an actor's competitiveness on the market, given an existing marketplace, or mechanism, it needs to adjust to. Each actor aims to maximize its own gains; any system of altruistic punishment inevitably falls outside the scope of this kind of simulation. For agent design, the input, i.e. the format of reputation information available from different actors in the marketplace, is fixed, but the actors themselves can choose the internal trust information models they follow themselves.

6.4.1 Reputation systems in electronic marketplaces

Related work literature presents multiple simulation experiments on the behavioural performance of different accumulative and probabilistic reputation systems in an electronic marketplace [SVB06, Nur07, JHF03]. In such a marketplace, intelligent agents, which correspond to our service providers, perform pairwise brief transactions of buying and selling goods. The marketplace is given a distribution of agents with different behaviour profiles, and each agent type has a decision policy; typically the reputation update policy is equal between all agents, and all experience information is shared. The simulation then measures for example the average number of transactions taken with a given type of agent (honest, malicious, etc).

The basic behaviour profiles of agents are typically very straightforward, such as “honest agents always carry out transactions honestly and give fair ratings”, while “malicious agents act honestly or dishonestly by chance, and always give negative ratings” [Nur07]. More complex behaviour can be tied to the marketplace as a whole; for example, a “spamming” agent can otherwise act honestly, but always rate other agents negatively in order to make itself more attractive in comparison [Nur07], or an agent may be an opportunistic defector, adjusting its behaviour based on whether there is anyone in the marketplace who will transact with it [JHF03]. Schlosser et al. define a behaviour profile for a “disturbing” agent as one who first builds a high reputation with good transactions, and then uses up the reputation so gained by defection [SVB06]; this represents the mechanism design equivalent of the optimal attacker model used in our simulations, which had an agent design focus.

When game-theoretic simulations are used like this for mechanism design, agent behaviour types are chosen globally for all policies under comparison, and measure when the system as a whole can or cannot protect the entire electronic marketplace. All agents, or at least all agents of a matching type, use the same decision algorithm, and if honest agents transact frequently with malicious agents, the reputation system has failed. Based on this definition, few reputation systems are resistant to the optimal attacker model — even the “disturbing” behaviour model [SVB06] turns out to be aptly named, when in fact it is nothing more than a model for a selfish agent behaving rationally within the limitations set by the environment.

The reason that observations are limited to fixed policies in these experiments is that in order to be able to give conclusive results, the tools of game theory require strict formalization of the environment and agent behaviour; the core problem then becomes how to formulate a question within this vocabulary so that it is "solvable", while ensuring that the result still gives some useful information about real marketplaces.

In order to make a successful reputation system for electronic commerce, we must aim to discourage the rational selfish agents from harmful behaviour, not just to protect unselfish "honest" agents from them. This is the social pressure effect of reputation systems, and it cannot be left outside our models: reputation systems that can only try to predict future behaviour are chronically powerless against intelligent adversaries that are able to change their behaviour to maximize their profit.

In our simulations, we have taken the agent design approach and study how a given agent survives on a marketplace with rational selfish agents. We have made no assumptions of what policies other actors use. We focus solely on the reputation update and decision policies of the single trustor, and the gains the single optimal attacker is able to make with an attack directed at the trustor using any of the different methods available to subvert the policies in place. By dropping these gains below a certain level we can discourage the rational attacker from misbehaviour. Protection against irrationally malicious agents is somewhat less likely to be needed, nor can it really be achieved through social control; for example random malfunctions do not depend on reputation, nor is a powerful group of outsiders trying to shut down the entire marketplace at any cost likely to care about their capability of operating on the market afterwards.

The main difference between electronic marketplace simulations and their metrics, and the simulations we have made in the previous sections, is in the point of view taken. In the aforementioned simulations, the simulator observes a marketplace of multiple actors to determine what the effect of the reputation mechanism has on the market as a whole. This also gives a nondeterministic flavour to the results: agents choose partners from the marketplace semi-randomly based on their reputation, their own behaviour rules may have some random elements (such as choosing to defect with probability p), and the distributions of different types of agent profiles on the marketplace may be expressed as probabilities rather than fixed numbers of each type.

In contrast, the TuBE simulations observe only two service actors at a time, use reputation for a binary trust decision rather than for partner selection, and have no random elements; the results are the same on each consecutive run. We do not enforce a single reputation system on the entire marketplace, and hence it also becomes less meaningful, in this context, to observe the combined effects of having a specific reputation system (or reputation update policy) in use by all actors.

In the simulations of Schlosser et al. [SVB06], the “disturbing” agent knows its reputation because the models have a centralized reputation system, i.e. the reputation update policy and experience information are shared between all agents. In comparison, Nurmi simulates a reputation system where the disturbing agent equivalent must actively estimate its own reputation due to not having access to any globally shared reputation value [Nur07]. It will have less information available than our optimal attacker, in favour of increased realism; on the other hand, a reputation model should not be relying on reputation information being kept secret either, so from a security perspective, the assumption of full knowledge is valuable.

In the TuBE model, policies and reputation information are kept private, which means that the normal attacker does not actually have any way of knowing when it should start defecting. In other words, it is separately empowered by the simulator in order to achieve optimal behaviour. This allows us to study the worst-case scenario from the point of view of defending against malicious agents.

Although electronic marketplace simulations seem to be a common approach for illustrating the behaviour of reputation systems, modelling attack cost and benefit for the attacker is not a new idea in evaluating resistance to a specific type of attack. The approach has also been adopted by for example Margolin et al. to quantify resistance to the Sybil attack, i.e. generating multiple identities to subvert a system by achieving a singlehanded majority [ML08], and somewhat differently by Srivatsa et al. in resisting oscillatory behaviour such as in the “disturbing” agent model mentioned above [SXL05].

The work by Margolin et al. is based on specific objectives of a rational attacker, and the expected cost of achieving them within one protection approach. Of the two, it is more similar to our work in philosophy, while the nature of attacks is quite different.

The TrustGuard system of Srivatsa et al., in turn, is more similar in its application area; central differences are that all transactions are equal in value and outcomes are binary. However, in this work, similarly to the prevalent electronic marketplace simulations, the attacker is not rational: it does not react to changes in its reputation, but rather changes its behaviour at fixed intervals. In other words, the attacker does not optimize its own behaviour for maximum benefit, but the trustor simply minimizes the gains of specific types of attackers within the cost-benefit model.

The difference between fixed and optimal attackers is small but significant in that the optimal approach involves finding the specific strategy that carries the greatest benefit to the attacker, and measuring its cost—in other words, all attack strategies will yield equal or worse outcomes for any kind of attacker. As we cannot choose our attackers in reality, we must assume a flexible rational attacker and measure the effectiveness of our defences against it.

As the optimal attacker strategy depends on the cost model chosen, the main challenge in evaluating the system is bound to finding a realistic cost model. In the work of Margolin et al., measuring cost is relatively easy: it is fixed to the cost of a new identity, which is fully determined by the system policy. The measurements then specify how highly the attacker must value a successful attack in order to choose to attempt it. Achieving this level of deterministic accuracy is less likely for reputation attacks, but as we have seen, it can still allow effective comparison of different policies within a chosen cost model. Developing a cost model for minimax analysis which can simultaneously model the losses from attacks, lost business due an overly strict decision policy and a form of ecosystem-enforced cost for failing to correctly punish misbehaviour is an interesting path of future research.

While the marketplace simulation approaches adopted in related work are reasonably well suited for studying statistical failures, i.e. reliability [Gol11], they are unsuited for evaluating the capability of specific actors to defend themselves against attacks. As a side effect of focusing on statistically modeled events in the marketplace as a whole, the simulated attackers do not behave rationally in these simulations, and as a result, the results do not reflect whether the chosen marketplace mechanism actually cuts down the incentive for a particular sort of attack. Our proposed approach of optimal attacker analysis is designed specifically for this attack deterrence point of view: how easy it is for an attacker to benefit from malicious behaviour. This kind of method could be applied to the marketplace as a whole as well, but would require considerable additional simplifications to model.

In this work, we apply our method locally for the purposes of agent design: selecting local policy of one agent so as to not be an attractive target for attacks. As we must respect the autonomy of all service providers, we cannot enforce a single decision policy upon all, and as a result, we can only assume that each agent acts primarily to further their own interests. In the next section, we contrast our approach to the state of the art in simulations focusing on the agent design point of view.

6.4.2 Competitive agents in the ART testbed

The Agent Reputation and Trust (ART) testbed initiative aims to establish a testbed for agent reputation and trust-related technologies [ART11]. It represents a competition-based approach to the evaluation of agent designs. The testbed serves both as a competition forum and an experimental tool. For further examples, the ART testbed specification lists similar, although generally simpler, experimentation environments [FKM+05].

The ART testbed simulates a marketplace of service providers competing to sell their services [FKM+05]. The provided service is art evaluation for a customer; in practice producing a real number as close to the unknown correct answer as possible. The service providers have a limited competency in the requested service, achieving different-quality results from their evaluations of art from specific eras. In addition, the correctness of the result depends on how much resources they decide to invest in producing the estimate themselves. They can also improve the quality of their own service by requesting help from other service providers that are more knowledgeable. The other providers have no innate interest in providing correct information, however. To exchange experiences, the actors can also ask each other for reputation information on other providers.

The testbed takes an important step forward in the work to provide benchmarking tools for trust and reputation systems. It is directed towards learning agents, with the goal to maximize their measured utility. In other words, each individual aims to maximize its own gains. The testbed specifies fixed prices for how much customers pay for an evaluation ($100), the cost of asking for an evaluation from another actor ($10), and the cost of asking for a reputation value (a real number between 0 and 1) from another actor ($1) [THD+07]. In addition, the agent can spend an arbitrary amount of money for its own evaluation, with the quality of information depending on the money spent. Teacy et al. provide further analysis of the ART testbed [THD+07, TCRJ08].

There are a few limitations that make ART insufficient as a benchmark environment for systems like TuBE. The testbed's background assumptions, low number (10-20) of actors in the marketplace, the uninformative reputation representation and the limited measurability of the quality of service are not compatible with the basic assumptions made in TuBE. Due to various factors in the testbed design, reputation information has actually not been particularly useful for an agent trying to perform well within the given bounds, and research focus has rather been misdirected towards secondary features of the game [GSMM08]. As an example, rather than use reputation information particularly wisely, the winning strategy directed its effort towards determining the most profitable amount of money to invest in opinions [THD$^+$07]. Discussing different ways to improve the testbed, Gomez et al. propose eliminating the use of money, and through it the power imbalance and secondary effects caused by it, and consider different approaches to make reputation information more valuable [GSMM08]. The testbed is currently unmaintained.

On a high level, the idea of making different policies compete in a marketplace fits the idea of measuring the competitiveness of different decision policy approaches very well: we can compare which specific policies help the agents applying them gain the most profit, and even study some emergent behaviour, such as whether the marketplace dies out as a result of too paranoid decision-making. The downside of this approach is that modelling a realistic marketplace is very different from modelling, say, an environment with realistic laws of physics. A marketplace economy is regulated to some degree by laws similar to those in natural sciences, but a large part of the outcome depends solely on subjective valuations, experience and interpretation, where there are no actual laws that can be deduced[4]. As noted in the evaluations of ART [THD$^+$07], we cannot conclude that an agent's competitiveness in the simulated marketplace necessarily has anything to do with the policy performing well for a real enterprise operating in a real marketplace.

When the service providers' goals and valuations are subjective, and their preferences unpredictable, what remains for a reasonable simulation target is to return to stereotypical actors, as applied in mechanism design. Instead of trying to second-guess what a service provider trustor would reasonably want to do, in our optimal attacker analysis for this work we pick a behaviour pattern that would generally not be desirable for a trustor to see a trustee exhibit in the marketplace: a collaboration partner deliberately violating contracts in a way that is clearly detrimental to the trustor.

We model this actor as an attacker whose internal rewards (its goals and valuations) are chosen to encourage contract violations, and set it out to maximize its rewards using rational selection within the set of actions made available to it. While this attacker may to a degree act as a reasonable

[4]This dual nature of the study of human action, or praxeology, is emphasized by e.g. von Mises in his work [vM98].

approximation of a real, antagonistic service provider with no intention to commit to a marketplace, it is important to realize that the goal of this artificial optimal attacker is not to represent a realistic actor as such, but a class of behaviour patterns that the *trustor*, i.e. the policy-setter, would by definition not want to attract or support. This is also why it is fruitful to give the attacker more power and knowledge than any realistic actor would have: it is used to evaluate how the trustor's defences succeed in increasing the cost of some attacks, not what drives a particular real attacker to be malicious.

In summary, we have identified common elements between our simulation and optimal attacker analysis and existing marketplace simulations for both mechanism design and agent design. We conclude that our approach, combining elements of the prevalent approaches and minimax analysis used in computer security evaluation, has more potential to produce fruitful results for policy-based systems.

6.5 Towards a benchmark for trust management systems

The experiments in this chapter have studied the role of simulations in the evaluation of trust management systems. Simulation experiments, like most measurement, are essentially methods of visualization. They are not proofs of the existence of desirable high-level characteristics, and definitely not tests for the system's resistance against anything but the specialized behaviour patterns chosen for simulations. As the ART testbed competitions show, even pitting different algorithms against each other in a testbed will teach us very little about their relative fitness in the world outside the testbed. Similarly, test loads from actual ecosystems, once they become available, remain selected datasets for visualization purposes.

What, then, should an automated test load, or "benchmark", consist of for a trust management system in the context of inter-enterprise collaborations? Given the above arguments, should the entire idea be abandoned as uninteresting? It is clear that the value of benchmarks lies outside attacker analysis, and relies on stereotypical behaviour patterns rather than realistic actors. We find that some characteristics are more fitting to demonstrate through simulations than others:

- **Constructive behaviour:** A simulation containing traces of reasonable behaviour, samples of "users want to do this", can be used to test that such behaviour is indeed supported by the system. Examples of interesting behaviour to simulate would include a few different types of reasonably reliable (albeit not perfect) service providers with different capabilities for service provision, or a newcomer with no reputation entering the ecosystem and finding partners that are willing to collaborate with it.

- **Efficiency:** For those parts of the systems that have non-trivial complexity, a more traditional benchmark load can be used to check the processing, network and storage load caused by decision-making and reputation processes. Different frequencies of decisions as well as incoming and outgoing reputation updates can be used to search for the limits of the scalability of the system. If no limits are found as the system handles all tested loads excellently, the load selection is too uninteresting for the system.

- **Recovery from problems:** A recovery simulation falls under the domain of reliability; it demonstrates how well the system tolerates expectable problems that can be modelled statistically. In the context of fixed loads, these include a service suddenly malfunctioning and becoming temporarily unreliable, a well-behaved user suffering and recovering from a defamation attempt against it (similarly to our stream 3 in the second experiment), or even a service being overloaded when its high reputation makes it too attractive to other actors in the ecosystem, for example.

Analyzing the gains of attackers is not included in the recovery from problems, even though they are one of the most important targets to study right after ensuring constructive behaviour towards benevolent actors. In the domain of security, attacks and defences form a continuous reaction loop, where new attacker attempts and new protections alternate. In comparison, simulating a non-rational, fixed-behaviour attacker holds very little interest, as it is more akin to malfunction patterns.

Malfunctions may produce an even stream of negative behaviour, and it is positive that a reputation system notices this change and eliminates the service until it recovers. However, it is important to note that in terms of providing a social control mechanism, the reputation system does not really deter random malfunctions; it simply detects them. A hope of good reputation can of course motivate a service provider to try to provide a more reliable service, but few providers will find it worth the investment to completely eliminate downtime for the purpose of gaining a spotless reputation.

On the other hand, the threat of reputation loss should deter deliberate attacks by making them more costly. For that to happen, attackers must be rational, aiming to maximize gains and minimize costs.

Rational attackers therefore form the final and particularly interesting target for analysis. Due to their dynamic behaviour, they are unsuitable for automated benchmarks, as they by necessity cannot be bound by fixed behaviour patterns. Rational attackers must choose their strategies based on what seems to work in and against a specific environment and policy setting. They should have a reasonable set of available strategies to choose from, together with defined cost and value for different actions and outcomes. In a contest-like setting, researchers could compete with two actors: an actor who gets things done while resisting attacks and recovering from problems, and an attacker approximating a "blind" attacker, who aims to defect in ways beneficial to itself against the other actors without full knowledge of the systems it tries to subvert. For a single system, however, it is rather more informative to analyze the optimal strategy for an attacker who has access to its reputation information. With this approach, we do not pollute the output with chance and assumptions about the difficulty of deducing the information.

As a dynamic attacker is poorly suited for simulated benchmarks, the attack resistance of different policies against such actors is better compared through minimax analysis, utilizing shared cost models and study of the resulting gained attacker utility points.

Rationality is not limited to attackers. Benevolent actors are also susceptible to different incentives created by a trust management system. Some of these incentives may encourage behavioural strategies that are unfortunate for the ecosystem, which should be considered when evaluating a system. For example, an important challenge raised in the context of reputation systems has been the lack of incentive to spontaneously share experiences.

A proposed solution has been to provide rewards for participation, but if rewards are based on quantity rather than quality, it creates an incentive to generate sloppy or random feedback as well. Similar issues lie in defining quality as a basis of rewards: if feedback quality is measured by defining statistical agreement with other feedback sources to be "correct" and rewardable, whistleblowers trying to alert the community of sudden misbehaviour will be punished rather than rewarded, and actual new and useful information is discouraged. Like optimal attackers, we find that these kinds of reward models must be analyzed for each system separately.

The objective reputation system proposed in Section 4.1.4 punishes actors for trying to defame their transaction partners as well as allows them to punish omission of positive experiences. This creates a contract-based incentive to submit experiences correctly, and also reduces the fear of retaliatory negative reports observed in electronic marketplaces [RZ02]. This approach provides a promising basis for a model of costs and benefits, where the three branches of attack resistance and risk aversion, desire to do business and maximize personal gains, and ecosystem sustainability through correct punishment of misbehaviour can all be taken into account.

6.6 Chapter summary

In this chapter, we have evaluated the trust management system based on six criteria: conceptual usability of trust aspects in decision making, support for autonomy, adjustability for different business situations, implementation of social control, scalability and feasibility, and attack resistance.

Related to the evaluation for adjustability and attack resistance, we have illustrated the behaviour of example trust decision and reputation update policies in connection with different experience streams. The compared policies have been chosen to represent different approaches to trust decisions and the reputation update process. Each policy forms its own balance between partially conflicting goals, such as avoiding information sparsity and accepting only quality information. As a result, the policies suit different environments and business needs, and demonstrate different levels of robustness against malicious behaviour.

We have also performed a game-theoretical analysis of the policies' resistance against optimal attackers, who have full knowledge of their own reputation in order to find the limits of the system's attack resistance. The policies performing best against these damage-maximizing attacks take advantage of central features in the TuBE information model: separation between minor and major effects of actions, and a measure of the credibility of the information source.

Finally, as a part of evaluating our evaluation methods, we have discussed the state of the art on simulation experiments in this field, and gauged the potential for building a benchmark for trust management in the inter-enterprise collaboration setting. We conclude that benchmarks could be set up for measuring some aspects of the collaboration, while actual attack resistance is best measured by other means. Our analysis of optimal attackers provides a new angle into this kind of evaluation in comparison to the prevalent methods in the field.

Chapter 7

Conclusion

Reputation-based trust management allows service providers to flexibly discover new partners for inter-enterprise collaborations, and react to changes in the behaviour of previously known partners. This thesis has presented a trust management system for automating routine trust decisions on joining and continuing in inter-enterprise collaborations. The trust decisions made by the system are based on continuously updated reputation information, and can be easily adjusted to different and changing business situations. The trust decisions are semi-automated, making an explicit division between routine cases, where automated decisions can be made with high certainty, and situations where the available information indicates a need for human intervention.

Trust decisions are based on an evaluation of the risks and benefits of a collaboration, as these are what constitutes the basis of rational decision-making. To capture the realistic considerations of the human decision-maker, we model these risks and benefits so that they cover different types of assets that are essential for the enterprise. This ensures that human users can accept and trust the system to make routine decisions for them. The TuBE trust management system implements an information model that provides better asset coverage than existing single-dimensional risk models, and allows the system to adjust to changes in the business situation as well as in the valuations of the enterprise.

Reputation-based trust management enforces social control in open service ecosystems. Without social control, service ecosystems are limited to closed communities of carefully selected and already familiar partners, while with the help of reputation systems, these ecosystems can evolve and grow in size more freely. The TuBE system combines both decisions based on shared experiences, which reward good behaviour and sanction misbehaviour across the ecosystem, and credibility evaluations of the shared experience information, in order to deter misbehaviour in reputation sharing.

7.1 Results

The TuBE trust management system combines the flexibility of policy-based decision systems with the learning capabilities of reputation-based systems. It can therefore be easily adjusted to different and changing business situations in inter-enterprise collaborations. The contributions of this work are fourfold: a multi-dimensional information model for computational trust decisions, the algorithms for making trust decisions privately within each enterprise, the aggregation algorithms for receiving, interpreting, evaluating and consuming experience information through reputation systems, and support for the enterprise to adjust to different business situations through private policies and metapolicies.

The trust information model we have presented extends the currently prevalent models in two directions. First, introducing asset-awareness into the model provides more expressive power in order to support meaningful risk estimations in a range of different business scenarios. Second, complementing the reputation factor with explicit trustor-dependent risk, importance, risk tolerance and context factors makes it possible to define flexible decision policies that reflect the valuations and needs of the enterprise, and use business-relevant concepts directly.

The processes for updating trust information and for producing trust decisions can be modified as required by local policy, at defined points. These points make it possible to easily adjust the behaviour of the trust management system. The basic algorithms are simple, understandable and scalable, and the separation between policy and implementation makes it possible to easily adjust the system to different and changing business situations. These attributes are essential in forming a system which enterprise users can rely on to make automated trust decisions for them.

The TuBE trust management system forms a part of the Pilarcos infrastructure for inter-enterprise collaboration management, and stands to gain significant supporting information from other management activities needed to establish and operate collaborations. With our prototype implementing the core trust management system and a set of example policies, we have been able to demonstrate that the system is feasible to implement, and the simulations show its adaptability in both trust decisions and reputation updating.

We have evaluated the trust management architecture based on six criteria: the conceptual usability of the trust decision making, support for autonomy, adjustability for different business situations, implementation of social control in the ecosystem, scalability and feasibility, and attack resistance. Through an analysis of the roles of different policies within the trust management system through the full arc of information processing and decisions, we have identified strengths and weaknesses in the example policies. These characteristics reflect the tradeoffs between for example fortifying the system against false reputation information, or being more accepting in order to have access to a broader input.

The results of this work can be utilized on three different levels. On the first level, we expect that small and medium-sized enterprises particularly can benefit from computational support to reduce the cost of collaboration management, enabling them to expand their networks beyond depending on a fixed contractor-subcontractor relationship. On the second level, we foresee that operators can benefit from offering services such as service offer storage and population, reputation information distribution or risk modelling of collaboration types. On the third level, entire domains of business, such as the forest industry, could benefit from the proposed infrastructure, including standardization of best practices in collaborations and reputation information, which allows very heterogeneous actors in the field to overcome their interoperability issues and enter into inter-enterprise collaborations.

In order to be applicable in practice for collaborations between individual enterprises, we have made two important design choices: First, we follow the principles of loose coupling and accept heterogeneity in order to ensure that the Pilarcos interoperability middleware and the trust management and reputation systems in it can be adopted without binding all enterprises under a single hub provider. Proposals based on everyone using the same system are unrealistic because enterprise size, assets and goals vary greatly. Consistent with this theme, TuBE can be configured to use the exact parts of the trust information model that bring value to a specific enterprise. The majority of solutions currently available focus on localized problems, and very few look for supporting information from backend infrastructure, such as TuBE gains from Pilarcos.

Second, we aim to preserve the autonomy of the participants, and instead design a degree of inherent distrust into the system in the form of continuous monitoring and trust decisions, because this allows service providers to collaborate without having to fully trust their collaborators. Enterprises that are competitors in one field can work together in another, for example, while they will naturally remain guarded about their privacy and internal business practices. This sets our work apart from e.g. traditional virtual enterprises and choreography-based collaboration models, which are based on more rigid pre-formed trust relationships, and hub actors running a collaboration over the other actors' systems.

7.2 Future work

Future work within TuBE can be split within three central goals: reputation interoperability, formal policy languages and trust information annotation within relevant modelling contexts.

Interoperability between reputation networks is essential in making relevant experience information available for trust decisions. TuBE acknowledges the need for multiple reputation systems, and is designed to consolidate information from several different systems. It is highly unlikely that the market should globally converge into a single reputation system, even if a sudden “killer app” of enterprise reputation information were to emerge at the right time. For this reason, reputation systems will have to be able to share information. The reputation information formats, processing and dissemination methods, the identities of trustees and recommenders, and the necessary division between first-hand and third-party information are all points of divergence between proposed systems [RKK07].

Interoperability is best supported by standardization. It is important that such a process is timed correctly; currently, the field is not yet mature enough for a credible proposal to have emerged for inter-enterprise collaborations. In the meanwhile, studying means of translating reputation information from one network to another forms a basis for using multiple reputation networks as information sources for the trust management system within an enterprise.

Another point of future work is in raising the abstraction level of policies further in order to merge trust management policies into the larger whole of collaboration management policies. Policies directing trust management must be expressed in a formal policy language in order to support configurability at runtime. In the current prototype, they are implemented as Java objects, which is a small, although significant, step away from fully hard-coded policies. This close to the implementation level, policy definition resembles scripting the trust management system, and therefore the language for these policies is less likely to be particularly useful outside the system. The information model also sets limits to what kinds of policies can be expressed.

Policy refinement makes it possible to provide higher-level configuration options using business concepts in one policy language, which is then translated to the more detailed and specialized policy language that can directly interface with the TuBE implementation. Selecting or designing suitable policy languages must be based on a requirement analysis of what kinds of things need to be possible to express.

Annotating trust information into relevant models of e.g. inter-enterprise collaboration and enterprise risk management makes the information available to automated processing. Producing the information relevant to trust management should not require a separate modelling round within each enterprise; it should be an integral part of modelling the situations trust information is needed in.

Modelling information relevant to trust decisions as a part of for example business network modelling makes it possible to enrich the trust decisions without having to manually feed in the same information in multiple places. Annotations in business network models should prove to be a useful first step in this direction.

References

[AH81] Axelrod, R. and Hamilton, W. D., The evolution of cooperation. *Science*, 211,4489(1981), pages 1390–1396. URL http://www.sciencemag.org/cgi/reprint/211/4489/1390.pdf.

[Alc10] Alcade, B., Trusted third party, who are you? *Short Paper Proceedings of the Fourth IFIP WG11.11 International Conference on Trust Management (IFIPTM 2010)*, Morioka, Iwate, Japan, jun 2010, pages 49–59, URL http://www.ifip-tm2010.org/lib/exe/fetch.php?media=shortpaper07.pdf.

[ARH00] Abdul-Rahman, A. and Hailes, S., Supporting trust in virtual communities. *Hawaii International Conference on System Sciences HICSS 2000*, January 2000, URL http://citeseer.ist.psu.edu/article/abdul-rahman00supporting.html.

[ART11] The agent reputation and trust (ART) testbed, August 2011. http://megatron.iiia.csic.es/art-testbed/index.html. [6.8.2010]

[Bay63] Bayes, T., An essay towards solving a problem in the doctrine of chances, by the late Rev. Mr. Bayes, F.R.S. communicated by Mr. Price in a letter to John Canton, A.M.F.R.S. *Philosophical Transactions of the Royal Society of London*, 53, pages 370–418. URL http://www.stat.ucla.edu/history/essay.pdf.

[BB05] Buchegger, S. and Boudec, J.-Y. L., Self-policing mobile ad hoc networks by reputation systems. *Communications Magazine*, pages 101–107. URL http://dx.doi.org/10.1109/MCOM.2005.1470831.

[BCE$^+$02] Bellwood, T., Clment, L., Ehnebuske, D., Hately, A., Hondo, M., Husband, Y. L., Januszewski, K., Lee, S., McKee, B., Munter, J. and von Riegen, C., *UDDI Version 3.0.*

UDDI Spec Technical Committee Specification, 19 July 2002. UDDI.org, July 2002. URL `http://uddi.org/pubs/uddi-v3.00-published-20020719.htm`.

[Bet10] Better Business Bureau, 2010. `http://www.bbb.org/`. [29.4.2008]

[BFK98] Blaze, M., Feigenbaum, J. and Keromytis, A. D., KeyNote: Trust management for public-key infrastructures (position paper). *Proceedings of Security Protocols: 6th International Workshop, Cambridge, UK, April 1998.* Springer-Verlag, LNCS 1550/1998, April 1998, pages 59–63, URL `http://www.springerlink.com/link.asp?id=6ku13fr5jt3mgdxk`.

[BFL96] Blaze, M., Feigenbaum, J. and Lacy, J., Decentralized trust management. *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, California, May 1996, IEEE, pages 164–173, URL `http://ieeexplore.ieee.org/iel3/3742/10940/00502679.pdf`.

[BHM$^+$04] Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C. and (Eds), D. O., Web Services architecture. W3C Working Group Note 11 February 2004. Technical Report, World Wide Web Consortium, February 2004. URL `http://www.w3.org/TR/ws-arch/`.

[BS04] Brændeland, G. and Stølen, K., Using risk analysis to assess user trust - a net-bank scenario. *Proceedings of Trust Management: Second International Conference, iTrust 2004, Oxford, UK, March 29–April 1, 2004.* Springer-Verlag, LCNS 2995/2004, March 2004, pages 146–160, URL `http://springerlink.metapress.com/openurl.asp?genre=article&issn=0302-9743&volume=2995&spage=146`.

[C$^+$03] Cahill, V. et al., Using trust for secure collaboration in uncertain environments. *Pervasive Computing*, 2,3(2003), pages 52–61. URL `http://ieeexplore.ieee.org/iel5/7756/27556/01228527.pdf`.

[CFL$^+$97] Chu, Y.-H., Feigenbaum, J., LaMacchia, B., Resnick, P. and Strauss, M., REFEREE: Trust management for Web applications. *Computer Networks and ISDN Systems*, 29,8–13(1997), pages 953–964. URL `http://dx.doi.org/10.1016/S0169-7552(97)00009-3`.

[CKK02] Clements, P., Kazman, R. and Klein, M., *Evaluating Software Architectures: Methods and Case Studies*. Addison Wesley, 2002.

[CW$^+$08] Carlsson, C., Weissmann, O. et al., AssessGrid D4.1: Advanced risk assessment, version 2.5. Technical Report, IST Information Society, Sixth Framework Programme, September 2008. URL `http://www.assessgrid.eu/fileadmin/AssessGrid/usermounts/publications/deliverables/ASSESSGRID_D4.1_Advanced_Risk_Assessment.pdf`.

[DDLS01] Damianou, N., Dulay, N., Lupu, E. and Sloman, M., The Ponder policy specification language. *Workshop on Policies for Distributed Systems and Networks (Policy2001), HP Labs Bristol, 29-31 Jan 2001*, volume 1995 of *LNCS*, January 2001, pages 18–38, URL `http://citeseer.ist.psu.edu/damianou01ponder.html`.

[DFM00] Dingledine, R., Freedman, M. and Molnar, D. *Accountability measures for peer-to-peer systems*, chapter 16. O'Reilly Publishers, 2000. URL `http://www.freehaven.net/doc/oreilly/accountability-ch16.html`.

[DLB08] Dondio, P., Longo, L. and Barrett, S., A translation mechanism for recommendations. In Karabulut, Y. et al. [KMHJ08], pages 87–102, URL `http://dx.doi.org/10.1007/978-0-387-09428-1_6`.

[Dou02] Douceur, J. R., The sybil attack. *Electronic Proceedings of the 1st International Workshop on Peer-to-Peer systems (IPTPS'02)*, Cambridge, MA, USA, March 2002, page 101, URL `http://www.cs.rice.edu/Conferences/IPTPS02/101.pdf`.

[eBa11] The eBay online marketplace, 2011. `http://www.ebay.com/`. [1.12.2008]

[Ell99] Ellison, C. M., The nature of a useable PKI. *Computer Networks*, 31, pages 823–830. URL `http://dx.doi.org/10.1016/S1389-1286(98)00018-8`.

[Ell04] Ellison, C. M., SPKI/SDSI certificate documentation, November 2004. `http://world.std.com/~cme/html/spki.html`. [31.7.2007]

[Eur07] European Commission, EC FP7 ICT Work Programme. Technical Report, EC, June 2007. URL `http://cordis.europa.eu/fp7/ict/`.

[EWN+03] English, C., Wagealla, W., Nixon, P., Terzis, S., McGettrick, A. and Lowe, H., Trusting collaboration in global computing systems. *First International Conference on Trust Management*, May 2003, pages 136–149, URL `http://springerlink.metapress.com/link.asp?id=yfhe8gg1398w088e`.

[FAB+06] Fitzgerald, B., Achatz, R., Bosch, J., Rombach, D., Beauvais, T., Fuggetta, A., Banâtre, J.-P., Banchilon, F., De Panfilis, S., Bomarius, F., Saikkonen, H., Kuilder, H., Boeckle, G. and Olsson, C. M., The software and services challenge. Technical Report, NESSI, January 2006. URL `ftp://ftp.cordis.europa.eu/pub/ist/docs/directorate_d/st-ds/fp7-report_en.pdf`.

[FF03] Fehr, E. and Fischbacher, U., The nature of human altruism. *Nature*, 425. URL `http://dx.doi.org/10.1038/nature02043`.

[FKM+05] Fullam, K. K., Klos, T. B., Muller, G., Sabater, J., Schlosser, A., Topol, Z., Barber, K. S., Rosenschein, J. S., Vercouter, L. and Voss, M., A specification of the Agent Reputation and Trust (ART) testbed: experimentation and competition for trust in agent societies. *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, 2005, pages 512–518, URL `http://doi.acm.org/10.1145/1082473.1082551`.

[FKÖD04] Fernandes, A., Kotsovinos, E., Östring, S. and Dragovic, B., Pinocchio: Incentives for honest participation in distributed trust management. *Proceedings of Trust Management: Second International Conference, iTrust 2004, Oxford, UK, March 29–April 1, 2004.* Springer-Verlag, LNCS 2995/2004, March 2004, pages 64–77, URL `http://springerlink.metapress.com/openurl.asp?genre=article&issn=0302-9743&volume=2995&spage=63`.

[GOGGF10] Gal-Oz, N., Grinshpoun, T., Gudes, E. and Friese, I., TRIC: An infrastructure for trust and reputation across virtual communities. *Proceedings of the Fifth International Conference on Internet and Web Applications and Services (ICIW 2010)*, Barcelona, Spain, May 2010, IEEE, pages 43–50, URL `http://doi.ieeecomputersociety.org/10.1109/ICIW.2010.14`.

[GOGH08] Gal-Oz, N., Gudes, E. and Hendler, D., A robust and knot-aware trust-based reputation model. In Karabulut, Y. et al.

[KMHJ08], pages 167–182, URL http://dx.doi.org/10.1007/978-0-387-09428-1_11.

[Gol11] Gollmann, D., From access control to trust management, and back — a petition. *Trust Management V; 5th IFIP WG 11.11 International Conference, IFIPTM 2011; Proceedings*, volume 358 of *IFIP AICT*, Copenhagen, Denmark, June/July 2011, Springer, pages 1–8.

[GS01] Grandison, T. W. and Sloman, M., Sultan - a language for trust specification and analysis. *Eighth Workshop of the HP OpenView University Association, Berlin, June 24-27, 2001.* HP OpenView University Association, June 2001. http://www.hpovua.org/PUBLICATIONS/PROCEEDINGS/8_HPOVUAWS/Papers/Paper01.2-Grandison-Sultan.pdf. [15.8.2005]

[GS02] Grandison, T. and Sloman, M., Specifying and analysing trust for Internet applications. *Proceedings of 2nd IFIP Conference on e-Commerce, e-Business, e-Government I3e2002, Lisbon, Portugal*, October 2002, URL http://citeseer.ist.psu.edu/grandison02specifying.html.

[GSMM08] Gomez, M., Sabater-Mir, J. and Muller, G., Improving the ART-testbed, thoughts and reflections. *Workshop on competitive agents in "Agent Reputation and Trust Testbed*, Salamanca, 2008, pages 1–15, URL http://megatron.iiia.csic.es/art-testbed/pdf/2007ArtCaepiaWS.pdf.

[Hei08] Heikkinen, S., Applicability of host identities to implement non-repudiable service usage. *International Journal on Advances in Systems and Measurements*, 1,1(2008), pages 14–28. URL http://www.iariajournals.org/systems_and_measurements/sysmea_v1_n1_2008_paged.pdf.

[HJS06a] Huynh, T. D., Jennings, N. R. and Shadbolt, N. R., Certified reputation: how an agent can trust a stranger. *Fifth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-06)*, Hakodate, Japan, May 2006, URL http://eprints.ecs.soton.ac.uk/12587/.

[HJS06b] Huynh, T. D., Jennings, N. R. and Shadbolt, N. R., An integrated trust and reputation model for open multi-agent systems. *Journal of Autonomous Agents and Multi-Agent Systems*, 13,2(2006), pages 119–154. URL `http://eprints.ecs.soton.ac.uk/12593/`.

[HKL$^+$04] Hull, R., Kumar, B., Lieuwen, D., Patel-Schneider, P. F., Sahuguet, A., Varadarajan, S. and Vyas, A., Enabling context-aware and privacy-conscious user data sharing. *Proceedings of the 2004 IEEE International Conference on Mobile Data Management (MDM'04)*, Berkeley, California, USA, 2004, IEEE, pages 187–198, URL `http://dx.doi.org/10.1109/MDM.2004.1263065`.

[Huh06] Huhns, M. N., A research agenda for agent-based service-oriented architectures. *Cooperative Information Agents X*, volume 4149 of *Lecture Notes in Computer Science*, 2006, pages 8–22, URL `http://dx.doi.org/10.1007/11839354_2`.

[INT05] INTEROP-NoE Task Group 7, Roadmap for TG7: Interoperability challenges of trust, confidence, security and policies. Technical Report, EU IST Information Society, 2005. Available through `http://interop-vlab.eu/`.

[INT07a] INTEROP-NoE Task Group 7, Deliverable DTG7.2 — Report on the horizontal issues and task group final report M36 including subtask research results following the five task group topics. Technical Report, EU IST Information Society, 2007. Available through `http://interop-vlab.eu/`.

[INT07b] INTEROP-NoE Task Group 7, Deliverable DTG7.3 — Research into non-functional aspects of interoperability. Technical Report, EU IST Information Society, 2007. Available through `http://interop-vlab.eu/`.

[ITU97] ITU/ISO, *ODP Trading Function – Part 1: Specification*, 1997. ITU/T Draft Rec X950–1.

[JH07] Jøsang, A. and Haller, J., Dirichlet reputation systems. *Proceedings of the Second International Conference on Availability, Reliability and Security (ARES 2007)*, Vienna, Austria, April 2007, IEEE Computer Society, pages 112–119, URL `http://dx.doi.org/10.1109/ARES.2007.71`.

[JHF03] Jøsang, A., Hird, S. and Faccer, E., Simulating the effect of reputation systems on e-markets. *Trust Management: First International Conference, iTrust 2003, Her-*

aklion, Crete, Greece, May 28–30, 2003. Proceedings, volume LNCS 2692/2003, May 2003, pages 179–194, URL `http://springerlink.metapress.com/openurl.asp?genre=article&issn=0302-9743&volume=2692&spage=179`.

[JI02] Jøsang, A. and Ismail, R., The beta reputation system. *Proceedings of the 15h Bled Electronic Commerce Conference*, Bled, Slovenia, June 2002, pages 324–337, URL `http://ecom.fov.uni-mb.si/proceedings.nsf/Proceedings/D9E48B66F32A7DFFC1256E9F00355B37/$File/josang.pdf`.

[JIB07] Jøsang, A., Ismail, R. and Boyd, C., A survey of trust and reputation systems for online service provision. *Decision Support Systems: Emerging Issues in Collaborative Commerce*, 43,2(2007), pages 618–644. URL `http://dx.doi.org/10.1016/j.dss.2005.05.019`.

[JMP06] Jøsang, A., Marsh, S. and Pope, S., Exploring different types of trust propagation. *Trust management*, LNCS 3986, 2006, pages 179–192, URL `http://folk.uio.no/josang/papers/JPM2006-iTrust.pdf`.

[JP04] Jøsang, A. and Presti, S. L., Analysing the relationship between risk and trust. *Proceedings of Trust Management: Second International Conference, iTrust 2004, Oxford, UK, March 29–April 1, 2004*. Springer-Verlag, LNCS 2995/2004, March 2004, pages 135–145, URL `http://springerlink.metapress.com/link.asp?id=mklyh19x5yb1c8n9`.

[JPBB04] Jung, J., Paxson, V., Berger, A. W. and Balakrishnan, H., Fast portscan detection using sequential hypothesis testing. *Proceedings of the IEEE Symposium on Security and Privacy*, May 2004, pages 211–225, URL `http://dx.doi.org/10.1109/SECPRI.2004.1301325`.

[Kar03] Karabulut, Y., Implementation of an agent-oriented trust management infrastructure based on a hybrid PKI model. *Proceedings of Trust Management: First International Conference, iTrust 2003, Heraklion, Crete, Greece, May 28–30, 2003.* Springer-Verlag, LNCS 2692/2003, May 2003, pages 318–331, URL `http://springerlink.metapress.com/openurl.asp?genre=article&issn=0302-9743&volume=2692&spage=318`.

[Kau11] Kaur, P., Users' trust decisions on inter-enterprise collaborations. Master's thesis, Aalto University, Computer Science and Engineering, September 2011.

[KBAW94] Kazman, R., Bass, L., Abowd, G. and Webb, M., Saam: A method for analyzing the properties of software architectures. *Proceedings of the 16th International Conference on Software Engineering*, Sorrento, Italy, 1994, pages 81–90.

[KBR05] Kinateder, M., Baschny, E. and Rothermel, K., Towards a generic trust model - comparison of various trust update algorithms. *Proceedings of Trust Management: Third International Conference, iTrust 2005, Paris, France, May 23–26, 2005*, Herrmann, P., Issarny, V. and Shiu, S., editors, volume 3477 of *LNCS*. Springer-Verlag, April 2005, pages 177–192, URL `http://www.springerlink.com/link.asp?id=qh7db1k6uqr3bl76`.

[KHKR06] Kerschbaum, F., Haller, J., Karabulut, Y. and Robinson, P., PathTrust: A trust-based reputation service for virtual organization formation. *Proceedings of Trust Management: 4th International Conference, iTrust 2006, Pisa, Italy, May 16–19, 2006*, volume 3986 of *LNCS*. Springer-Verlag, May 2006, pages 193–205, URL `http://dx.doi.org/10.1007/11755593_15`.

[KKC00] Kazman, R., Klein, M. and Clements, P., ATAM: Method for architecture evaluation. Technical Report CMU/SEI-2000-TR-004, ESC-TR-2000-004, Carnegie Mellon Software Engineering Institute, August 2000. URL `http://www.sei.cmu.edu/reports/00tr004.pdf`.

[KLMR07] Kutvonen, L., Linington, P., Morin, J.-H. and Ruohomaa, S., editors, *Pre-proceedings of IS-TSPQ 2007 — The 2nd international workshop on Interoperability solutions to Trust, Security, Policies and QoS for Enhanced Enterprise Systems*. University of Helsinki, Department of Computer Science Publications Series B, Report B-2007-3, March 2007.

[KMHJ08] Karabulut, Y., Mitchell, J., Herrmann, P. and Jensen, C. D., editors. *Trust Management II*, volume 263 of *IFIP International Federation for Information Processing*, Pisa, Italy, May 2008. Springer.

[KMR05] Kutvonen, L., Metso, J. and Ruokolainen, T., Inter-enterprise collaboration management in dynamic business networks. *On* *the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE*, volume 3760 of *Lecture Notes in Computer Science*, Agia Napa, Cyprus, November 2005, Springer-Verlag, pages 593–611, URL http://dx.doi.org/10.1007/11575771_37.

[KMR07] Kutvonen, L., Metso, J. and Ruohomaa, S., From trading to eCommunity management: Responding to social and contractual challenges. *Information Systems Frontiers (ISF) - Special Issue on Enterprise Services Computing: Evolution and Challenges*, 9,2–3(2007), pages 181–194. URL http://dx.doi.org/10.1007/s10796-007-9031-x.

[KP03] Kinateder, M. and Pearson, S., A privacy-enhanced peer-to-peer reputation system. *Proceedings of E-Commerce and Web Technologies: 4th International Conference, EC-Web Prague, Czech Repbulic, September 2-5, 2003*, volume 2738 of *LNCS*. Springer-Verlag GmbH, September 2003, pages 206–215, URL http://www.springerlink.com/link.asp?id=yd15xa1pf68g9jmx.

[KR03] Kinateder, M. and Rothermel, K., Architecture and algorithms for a distributed reputation system. *Proceedings of Trust Management: First International Conference, iTrust 2003, Heraklion, Crete, Greece, May 28-30, 2003*, Nixon, P. and Terzis, S., editors, volume 2692 of *LNCS*. Springer-Verlag, May 2003, pages 1–16, URL http://www.springerlink.com/link.asp?id=1e11b5pcvww3qvuf.

[KRM07] Kutvonen, L., Ruokolainen, T. and Metso, J., Interoperability middleware for federated business services in web-Pilarcos. *International Journal of Enterprise Information Systems, Special issue on Interoperability of Enterprise Systems and Applications*, 3,1(2007), pages 1–21. URL http://www.idea-group.com/articles/details.asp?id=6597.

[KRM08] Kutvonen, L., Ruohomaa, S. and Metso, J., Automating decisions for inter-enterprise collaboration management. *Proceedings of the 9th IFIP Working Conference on Virtual Enterprises (PRO-VE 2008)*, Poznan, Poland, September 2008, URL http://www.cs.helsinki.fi/group/cinco/publications/public_pdfs/kutvonen08automating.pdf.

[KRRM08] Kutvonen, L., Ruokolainen, T., Ruohomaa, S. and Metso, J., Service-oriented middleware for managing inter-enterprise collaborations. *Global Implications of Modern Enterprise Information Systems: Technologies and Applications*, Advances in Enterprise Information Systems (AEIS). IGI Global, December 2008, pages 209–241, URL `http://www.igi-global.com/reference/details.asp?id=9648`.

[KSGM03] Kamvar, S., Schlosser, M. and Garcia-Molina, H., The EigenTrust algorithm for reputation management in P2P networks. *Proceedings of the Twelfth International World-Wide Web Conference (WWW03)*, Budapest, Hungary, May 2003, pages 446–458, URL `http://www2003.org/cdrom/papers/refereed/p446/p446-kamvar/index.html`.

[Kut02] Kutvonen, L., Automated management of interorganisational applications. *Sixth International Enterprise Distributed Object Computing Conference (EDOC '02)*, Lausanne, Switzerland, September 2002, IEEE, pages 27–38, URL `http://www.cs.helsinki.fi/group/pilarcos/deliverables/kutvonen_management_edoc_2002.pdf`.

[Lam81] Lamport, L., Password authentication with insecure communication. *Communications of the ACM*, 24. URL `http://dx.doi.org/10.1145/358790.358797`.

[LCDP06] Li, M.-S., Cabral, R., Doumeingts, G. and Popplewell, K., Enterprise interoperability. Research Roadmap. Technical Report, EC, July 2006. URL `http://cordis.europa.eu/ist/ict-ent-net/ei-roadmap_en.htm`.

[LdBSV04] Lund, M. S., den Braber, F., Stølen, K. and Vraalsen, F., An UML profile for the identification and analysis of security risks during structured brainstorming. Technical Report, SINTEF ICT, May 2004. URL `http://coras.sourceforge.net/documents/uml-sa-report2.pdf`.

[LM10] Li, Q. and Martin, K. M., A secure marketplace for online services that induces good conduct. *Short Paper Proceedings of the Fourth IFIP WG11.11 International Conference on Trust Management (IFIPTM 2010)*, Morioka, Iwate, Japan, jun 2010, pages 65–72, URL `http://www.ifip-tm2010.org/lib/exe/fetch.php?media=shortpaper09.pdf`.

[LMS+07] Lysemose, T., Mahler, T., Solhaug, B., Bing, J., Elgesem, D. and Stølen, K., ENFORCE conceptual framework. Technical Report, SINTEF ICT, 2007. URL `http://www.uib.no/People/bso002/reports/A1209_enforceCF.pdf`.

[LS+05] Lutz Schubert, M. W. et al., TrustCoM reference architecture, Deliverable D09. Technical Report, TrustCoM WP27, August 2005. URL `http://www.eu-trustcom.com/DownDocumentation.php?tipo=docu&id=208`.

[LSB03] Lee, S., Sherwood, R. and Bhattacharjee, B., Cooperative peer groups in NICE. *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003)*, volume 2. IEEE, April 2003, pages 1272–1282, URL `http://ieeexplore.ieee.org/iel5/8585/27206/01208963.pdf`.

[M+06] MacKenzie, C. M. et al., Oasis reference model for service oriented architecture 1.0. Technical Report, OASIS, July 2006. URL `http://www.oasis-open.org/committees/download.php/19361/soa-rm-cs.pdf`.

[MA07] Msanjila, S. S. and Afsarmanesh, H., HICI: An approach for identifying trust elements — the case of technological trust perspective in VBEs. *Proceedings of the Second International Conference on Availability, Reliability and Security (ARES 2007)*, Vienna, Austria, April 2007, IEEE Computer Society, pages 757–764, URL `http://dx.doi.org/10.1109/ARES.2007.94`.

[MAH+06] Msanijla, S. S., Afsarmanesh, H., Hodik, J., Rehk, M. and Camarinha-Matos, L. M., ECOLEAD deliverable D21.4b: Creating and supporting trust culture in VBEs. Technical Report, EC Information Society, March 2006. URL `http://www.ve-forum.org/projects/284/Deliverables/D21.4b_Final.pdf`.

[MC96] McKnight, D. H. and Chervany, N. L., The meanings of trust. Technical Report, University of Minnesota, MIS Research Center, 1996. URL `http://misrc.umn.edu/workingpapers/fullPapers/1996/9604_040100.pdf`.

[MG06] Mens, T. and Gorp, P. V., A taxonomy of model transformation. *Proceedings of the International Workshop on Graph and Model Transformation (GraMoT 2005)*, volume 152 of *Electronic Notes in Theoretical Computer Science*. Elsevier, March 2006, pages 125–142, URL http://dx.doi.org/10.1016/j.entcs.2005.10.021.

[MG10] Mehandiev, N. and Grefen, P., editors, *Dynamic Business Process Formation for Instant Virtual Enterprises.* Advanced Information and Knowledge Processing. Springer, June 2010.

[ML08] Margolin, N. B. and Levine, B. N., Quantifying resistance to the Sybil attack. *Proceedings of Financial Cryptography and Data Security (FC 2008)*, Cozumel, Mexico, January 2008, Springer, pages 1–15, URL http://dx.doi.org/10.1007/978-3-540-85230-8_1.

[MMH02] Mui, L., Mohtashemi, M. and Halberstadt, A., A computational model of trust and reputation. *35th Annual Hawaii International Conference on System Sciences (HICSS'02)*, volume 7. IEEE Computer Society, January 2002, page 188, URL http://doi.ieeecomputersociety.org/10.1109/HICSS.2002.994181.

[MRVK10] Moen, P., Ruohomaa, S., Viljanen, L. and Kutvonen, L., Safeguarding against new privacy threats in inter-enterprise collaboration environments. Technical Report C-2010-56, University of Helsinki, Department of Computer Science, 2010.

[Nat10] National consumer agency of Ireland, 2010. http://www.consumerconnect.ie/. [14.1.2010]

[NDA$^+$07] Nachira, F., Dini, P., A.Nicolai, Louarn, M. and Léon, L., *Digital Business Ecosystems.* European Commission, 2007. URL http://www.digital-ecosystems.org/book/de-book2007.html.

[NPC$^+$04] Norman, T. J., Preece, A. D., Chalmers, S., Jennings, N. R., Luck, M., Dang, V. D., Nguyen, T. D., Deora, V., Shao, J., Gray, W. A. and Fiddian, N. J., Agent-based formation of virtual organisations. *Knowledge-based systems*, 17,2–4(2004), pages 103–111. URL http://dx.doi.org/10.1016/j.knosys.2004.03.005.

[Nur06] Nurmi, P., A Bayesian framework for online reputation systems. *Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services AICT/ICIW '06).* IEEE Computer Society, February 2006, page 121, URL http://dx.doi.org/10.1109/AICT-ICIW.2006.2.

[Nur07] Nurmi, P., Perseus – a personalized reputation system. *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*. IEEE Computer Society, 2007, pages 798–804, URL `http://dx.doi.org/10.1109/WI.2007.121`.

[OAS07] OASIS Web Service Secure Exchange TC, *WS-Trust 1.3 OASIS Standard*. OASIS, March 2007. URL `http://docs.oasis-open.org/ws-sx/ws-trust/v1.3/ws-trust.pdf`.

[Obr04] Obreiter, P., A case for evidence-aware distributed reputation systems overcoming the limitations of plausibility considerations. *Proceedings of Trust Management: Second International Conference, iTrust 2004, Oxford, UK, March 29–April 1, 2004*. Springer-Verlag, LNCS 2995/2004, March 2004, pages 33–47, URL `http://www.springerlink.com/link.asp?id=ucc149f215dhyj0y`.

[Pap03] Papazoglou, M. P., Service-oriented computing: Concepts, characteristics, and directions. *Proceedings of the 4th International Conference on Web Information Systems Engineering (WISE 2003)*. IEEE Computer Society, December 2003, URL `http://infolab.uvt.nl/pub/papazogloump-2003-51.pdf`.

[PTD$^+$06] Papazoglou, M. P., Traverso, P., Dustdar, S., Leymann, F. and Krämer, B. J., Service-oriented computing: A research roadmap. *Service Oriented Computing (SOC)*, Cubera, F., Krämer, B. J. and Papazoglou, M. P., editors, number 05462 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2006, Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, URL `http://drops.dagstuhl.de/opus/volltexte/2006/524/`.

[PTJ$^+$06] Patel, J., Teacy, W. T. L., Jennings, N. R., Luck, M., Chalmers, S., Oren, N., Norman, T. J., Preece, A. D., Gray, P. M. D., Shercliff, G., Stockreisser, P. J., Shao, J., Gray, W. A., Fiddian, N. J. and Thompson, S. G., CONOISE-G: agent-based virtual organisations. *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006)*, Hakodate, Japan, 2006, ACM, pages 1459–1460, URL `http://doi.acm.org/10.1145/1160633.1160914`.

[RGAN06] Rabelo, R. J., Gusmeroli, S., Arana, C. and Nagellen, T., The ECOLEAD ICT infrastructure for collaborative networked organizations. *Network-Centric Collaboration and Supporting Frameworks*, volume 224. Springer, 2006, pages 451–460.

[RHK11] Ruohomaa, S., Hankalahti, A. and Kutvonen, L., Detecting and reacting to changes in reputation flows. *Trust Management V*, volume 358 of *IFIP Advances in Information and Communication Technology*, Copenhagen, Denmark, June 2011, pages 19–34, URL http://dx.doi.org/10.1007/978-3-642-22200-9_5.

[RJ96] Rasmusson, L. and Jansson, S., Simulated social control for secure Internet commerce. *Proceedings of the 1996 workshop on New Security Paradigms*. ACM Press, 1996, pages 18–25, URL http://doi.acm.org/10.1145/304851.304857.

[RK05] Ruohomaa, S. and Kutvonen, L., Trust management survey. *Proceedings of the iTrust 3rd International Conference on Trust Management, 23–26, May, 2005, Rocquencourt, France*, volume 3477 of *Lecture Notes in Computer Science*. Springer-Verlag, May 2005, pages 77–92, URL http://dx.doi.org/10.1007/11429760_6.

[RK06] Ruohomaa, S. and Kutvonen, L., Luottamuksenhallinta avoimissa palveluverkoissa. *Tietojenkäsittelytiede*, 25, pages 51–60. In Finnish.

[RK07] Ruokolainen, T. and Kutvonen, L., Service Typing in Collaborative Systems. *Enterprise Interoperability: New Challenges and Approaches*, Doumeingts, G., Müller, J., Morel, G. and Vallespir, B., editors. Springer, April 2007, pages 343–354.

[RK10] Ruohomaa, S. and Kutvonen, L., Trust and distrust in adaptive inter-enterprise collaboration management. *Journal of Theoretical and Applied Electronic Commerce Research*, 5,2(2010), pages 118–136. URL http://www.jtaer.com/aug2010/ruohomaa_kutvonen_p7.pdf.

[RK11] Ruohomaa, S. and Kutvonen, L., From subjective reputation to verifiable experiences - implementing social control in open service ecosystems. Technical Report, University of Helsinki, Department of Computer Science, 2011. Internal technical report.

[RKK07] Ruohomaa, S., Kutvonen, L. and Koutrouli, E., Reputation management survey. *Proceedings of the 2nd International Conference on Availability, Reliability and Security (ARES 2007)*, Vienna, Austria, April 2007, IEEE Computer Society, pages 103–111, URL http://dx.doi.org/10.1109/ARES.2007.123.

[RN03] Russell, S. and Norvig, P. *Artificial Intelligence — A Modern Approach*, chapter 6: Adversarial search. Prentice Hall, second edition, 2003.

[RRRJ07a] Reece, S., Roberts, S., Rogers, A. and Jennings, N. R., A multi-dimensional trust model for heterogeneous conract observations. *Twenty-second AAAI conference on artificial intelligence*, Vancouver, Canada, July 2007, URL http://eprints.ecs.soton.ac.uk/13867/.

[RRRJ07b] Reece, S., Rogers, A., Roberts, S. and Jennings, N. R., Rumours and reputation: evaluating multi-dimensional trust within a decentralized reputation system. *The sixth international joint conference on autonomous agents and multi-agent systems (AAMAS-07)*, Honolulu, Hawaii, USA, May 2007, pages 1063–1070, URL http://eprints.ecs.soton.ac.uk/13260/.

[RSS08] Refsdahl, A., Solhaug, B. and Stølen, K., An UML-based method for the development of policies to support trust management. In Karabulut, Y. et al. [KMHJ08], pages 33–50, URL http://dx.doi.org/10.1007/978-0-387-09428-1_3.

[Ruo06] Ruohomaa, S., Trust management concepts and methodology. *Proceedings of FDPW'2005—Advances in Methods of Modern Information Technology*, volume 7. Petrozavodsk State University, 2006, pages 180–193.

[RZ02] Resnick, P. and Zeckhauser, R., Trust among strangers in internet transactions: Empirical analysis of eBay's reputation system. *The Economics of the Internet and E-Commerce*, volume 11 of *Advances in Applied Microeconomics*. Elsevier Science, Amsterdam, 2002, pages 127–157, URL http://www.si.umich.edu/~presnick/papers/ebayNBER/RZNBERBodegaBay.pdf.

[RZFK00] Resnick, P., Zeckhauser, R., Friedman, E. and Kuwabara, K., Reputation systems. *Communications of the ACM*, 43,12(2000), pages 45–48. URL http://doi.acm.org/10.1145/355112.355122.

[Sch06] Schmidt, D. C., Model-driven engineering. *IEEE Computer*, 39, pages 25–31. URL http://www.cs.wustl.edu/~schmidt/GEI.pdf.

[SES07] Solhaug, B., Elgesem, D. and Stølen, K., Specifying policies using UML sequence diagrams: An evaluation based on a case study. *Proceedings of the Eigth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'07)*. IEEE Computer Society, June 2007, pages 19–28, URL http://dx.doi.org/10.1109/POLICY.2007.42.

[SFJ+03] Seigneur, J.-M., Farrell, S., Jensen, C. D., Gray, E. and Yong, C., End-to-end trust starts with recognition. Technical Report, Trinity College Dublin, 2003. URL http://www.cs.tcd.ie/publications/tech-reports/reports.03/TCD-CS-2003-05.pdf.

[Sol06] Solove, D. J., A taxonomy of privacy. *University of Pennsylvania Law Review*, 154,3(2006), pages 477–560. URL http://ssrn.com/abstract=667622. GWU Law School Public Law Research Paper No. 129.

[Sol09] Solhaug, B., *Policy Specification Using Sequence Diagrams. Applied to Trust Management.* Ph.D. thesis, University of Bergen, Bergen, Norway, October 2009. URL https://bora.uib.no/handle/1956/3723.

[SS02] Sabater, J. and Sierra, C., Reputation and social network analysis in multi-agent systems. *AAMAS '02: Proceedings of the First International Joint Conference on Autonomous Agents and MultiAgent Systems*, Bologna, Italy, 2002, pages 475–482, URL http://doi.acm.org/10.1145/544741.544854.

[Sta10] Standard & Poor's website, 2010. http://www.standardandpoors.com. [25.7.2007]

[SVB06] Schlosser, A., Voss, M. and Brückner, L., On the simulation of global reputation systems. *Journal of Artificial Societies and Social Simulation*, 9,1(2006). URL http://jasss.soc.surrey.ac.uk/9/1/4/4.pdf.

[SXL05] Srivatsa, M., Xiong, L. and Liu, L., TrustGuard: countering vulnerabilities in reputation management for decentralized overlay networks. *WWW '05: Proceedings of the 14th International*

Conference on the World Wide Web, New York, USA, May 2005, ACM Press, pages 422–431, URL http://doi.acm.org/10.1145/1060745.1060808.

[TCRJ08] Teacy, W. L., Chalkiadakis, G., Rogers, A. and Jennings, N. R., Sequential decision making with untrustworthy service providers. *Proceedings of the 7th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2008)*, Estoril, Portugal, 2008, URL http://eprints.ecs.soton.ac.uk/15168/.

[THD+07] Teacy, W. L., Huynh, T. D., Dash, R. K., Jennings, N. R., Luck, M. and Patel, J., The ART of IAM: The winning strategy for the 2006 competition. *Proceedings of the AAMAS Workshop on Trust in Agent Societies*, Hawaii, USA, 2007, URL http://eprints.ecs.soton.ac.uk/13718/.

[The05] The Athena project, Deliverable D.A2.1: Cross-organisational business process requirements and the state of the art in research, technology and standards; version 2.0. Technical Report, Athena project, November 2005. URL http://interop-vlab.eu/ei_public_deliverables/athena-deliverables/A2/d-a2.1.

[TPJL06] Teacy, W. T. L., Patel, J., Jennings, N. R. and Luck, M., TRAVOS: Trust and reputation in the context of inaccurate reputation sources. *Autonomous Agents and Multi-agent Systems*, 12,2(2006), pages 183–198. URL http://www.springerlink.com/content/2h56k13n37qk0274/.

[Tru10] Trustmark tradesman certification, 2010. http://www.trustmark.org.uk/. [14.1.2010]

[UBJ04] Uszok, A., Bradshaw, J. M. and Jeffers, R., KAoS: A policy and domain services framework for grid computing and Semantic Web services. *Proceedings of Trust Management: Second International Conference, iTrust 2004, Oxford, UK, March 29–April 1, 2004*, volume Springer-Verlag, LNCS 2995/2004, March 2004, pages 16–26, URL http://dx.doi.org/10.1007/b96545.

[Vil05a] Viljanen, L., A survey on application level intrusion detection. Technical Report, University of Helsinki, Department of Computer Science, 2005.

[Vil05b] Viljanen, L., Towards an ontology of trust. *Trust, Privacy, and Security in Digital Business. Second International Conference,* *TrustBus 2005*, volume 3592 of *Lecture Notes in Computer Science*, Copenhagen, Denmark, 2005, Springer-Verlag, pages 175–184, URL `http://dx.doi.org/10.1007/11537878_18`.

[Vil11] Viljanen, L., Trust and mistrust management in enterprise systems, 2011. Licentiate thesis, University of Helsinki, Department of Computer Science.

[vLAV05] von Laszewski, G., Alunkal, B. and Velikovic, I., Toward reputable Grids. *Scalable Computing: Practice and Experience*, 6,3(2005). URL `http://www.scpe.org/vols/vol06/no3/SCPE_6_3_09.pdf`.

[vM98] von Mises, L., *Human Action — A Treatise on Economics; The Scholar's Edition.* Ludvig von Mises Institute, Auburn, Alabama, US, 1998. URL `http://mises.org/resources/3250`.

[W$^+$06] Wilson, M. et al., The TrustCoM approach to enforcing agreements between interoperating enterprises. *Enterprise Interoperability. New Challenges and Approaches*, Bordeaux, France, March 2006, Springer-Verlag, pages 365–376, URL `http://epubs.cclrc.ac.uk/bitstream/898/Trustcom_Interoperability_France.pdf`.

[Wal45] Wald, A., Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16,2(1945), pages 117–186. URL `http://dx.doi.org/10.1214%2Faoms%2F1177731118`.

[WCE$^+$03] Wagealla, W., Carbone, M., English, C., Terzis, S. and Nixon, P., A formal model on trust lifecycle management. In *Workshop on Formal Aspects of Security and Trust (FAST2003) at FM2003*, volume IIT TR-10/2003, IIT-CNR, Italy, September 2003, pages 184–195. URL `http://www.iit.cnr.it/FAST2003/fast-proc-final.pdf` (TR-10/2003).

[Win03] Winslett, M., An introduction to trust negotiation. *Trust Management: First International Conference, iTrust 2003, Heraklion, Crete, Greece, May 28–30, 2003. Proceedings*, volume LNCS 2692/2003, May 2003, pages 275–283, URL `http://www.springerlink.com/link.asp?id=ufuj17106khufcj2`.

[WSJ00] Winsborough, W. H., Seamons, K. E. and Jones, V. E., Automated trust negotiation. *Proceedings of the DARPA Information Survivability Conference and Exposition DISCEX '00*, volume 1.

IEEE, January 2000, pages 88–102, URL `http://ieeexplore.ieee.org/iel5/6658/17862/00824965.pdf`.

[YIP$^+$00] Yavatkar, R., Intel, Pendarakis, D., IBM, Guerin, R. and U. of Pennsylvania, *A Framework for Policy-based Admission Control*, January 2000. URL `http://www.ietf.org/rfc/rfc2753.txt`.

[YRX12] Yao, Y., Ruohomaa, S. and Xu, F., Addressing common vulnerabilities of reputation systems for electronic commerce. *Journal of Theoretical and Applied Electronic Commerce Research*, 7, pages 1–15. To appear.

[YS03] Yu, B. and Singh, M. P., Searching social networks. *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, Melbourne, Australia, 2003, ACM, pages 65–72, URL `http://doi.acm.org/10.1145/860575.860587`.

Appendix I

Glossary

This glossary defines terms related to the Pilarcos middleware and TuBE trust management system in particular. Some of the terms are not used in this thesis directly, but appear repeatedly in earlier publications.

action: The activities in a collaboration are divided into actions, and a trust decision is made for each action rather than only once for the entire collaboration. On a technical level, an action is represented by the exchange of one or more messages between the service provider and user, and each action has a point where the commitment requiring a trust decision is made. See also task.

actor: The actors (trustors and trustees) in TuBE are business services. Services are globally identifiable, while for example the enterprise offering a service may in fact be an entire business network. This said, if reputation information is provided of an enterprise, it can be propagated into the services, and similarly reports made to that network based on the behaviour of the services of a given service provider. This conversion is done at the border of the reputation network, however, and remains invisible to the rest of the trust management system.

asset: An asset is an abstract or concrete resource an enterprise aims to protect and strengthen in a collaboration. In TuBE, the wide range of different assets is represented by high-level concepts; for example, any asset with a simply defined monetary value is considered a part of the monetary asset in decision-making. The four defined TuBE assets are monetary, reputation, control and fulfilment (also denoted as satisfaction in later publications to allow a broader interpretation than fulfilment of agreements). Other, more specific or completely different assets can be added to a local TuBE implementation, but at the cost of reduced interoperability in experience sharing.

base risk: A part of the implementation of risk evaluation in TuBE, the base risk is a function to trim the probabilities of impossible outcomes from a generic vector of probabilities, which is calculated based on reputation. It is formally defined in Section 3.2.

business-to-business (B2B) collaboration: In Pilarcos, business-to-business collaboration is used synonymously to inter-enterprise collaboration.

business network: A business network is the concrete instance of a business network model; to form a business network, service providers have been assigned into their respective roles defined in the model by a populator. A business network consists of the service providers, represented by their service offers selected for the network; the activity taking place in an operational business network is referred to as an inter-enterprise collaboration.

business network initiator: A business network initiator is a service provider who wishes to set up a business network. The initiator calls the populator service to find suitable partners to fill the roles in a business network model, and directs the contract negotiation process.

business network model: A business network model (BNM) defines the roles of a business network, interactions between them, and policy on how the network is supposed to operate, for example the allowed orderings of messages [KRRM08].

business service: A business service is the combination of a computational service application with monitoring facilities, which enforce local policy. The computational service can have capabilities for varying behaviour, not all of which are desirable in all contexts, while the monitor can be configured to allow only certain subsets of the behaviour in a given operating context, such as within a collaboration or towards a specific actor. See also service.

certainty: Certainty is a measure of the quality of information used for a trust decision, representing the trustor's confidence in that a decision based on the information will be well-informed. While credibility focuses specifically on the expected accuracy of information from external sources, certainty also measures the amount of information available. In the literature, certainty is occasionally referred to via its opposite, uncertainty [C^+03], to emphasize that trust decisions are always based on estimates and incomplete information.

CINCO: The Collaborative and Interoperable Computing (CINCO) group is an umbrella project under which the different parts of the Pilarcos middleware, including the TuBE trust management system, are developed at University of Helsinki.

cold start problem: See newcomer.

collaboration: An inter-enterprise collaboration involves a group of enterprises in a contractually regulated, cooperative project. The technical implementation of a collaboration is referred to as a business network.

collaboration epoch: In Pilarcos, an epoch refers to a phase in a collaboration. The behaviour of the business network may vary between epochs; for example, when a supply-chain type of collaboration moves from preparations to actual production and on towards dissolution, different kinds of activities become relevant. Epochs are defined in the business network model. See also reputation epoch.

collaboration management: Collaboration management refers to activities to support setting up, operating and terminating a collaboration. Pilarcos is a collaboration management middleware, providing support for setting up collaborations, runtime monitoring and breach management, and dissolution, including the exchange of experience information.

computational service: A computational service application provides the functionality of a business service. See also service.

cooperation: Fulfilling an agreement. The terms of cooperation and defection have been adopted from game theory, where in a famous example of the Prisoner's Dilemma game [AH81] two imprisoned culprits' options are limited to testifying for the prosecution to receive a shorter sentence themselves (defect) or remaining silent in the hopes that the other does as well, in which case both get away on a minor charge (cooperate). In a game-theoretical sense, a cooperating actor fulfils its duties, in the hopes of a greater mutual gain. This includes an assumption that the other actor cooperates as well; otherwise the cooperating actor typically suffers losses, while the defecting party is the only one to gain. See also defection.

contract: In this thesis, contract is used synonymously to eContract; see also eContract.

control: Control is a TuBE asset used to represent the enterprise's ability to protect itself from outside influences and to maintain its security, privacy and other aspects of its autonomy. The control asset is based on the autonomy and self-control of the enterprise.

credibility: Credibility is a form of information source "trustworthiness" and applicability in a given context. In TuBE, external reputation information is assigned a credibility value to indicate how strongly the trustor subjectively believes the source to be accurate and useful; the credibility value does not indicate the accuracy of the particular item of information. When referring to the credibility of external experience information, we mean the credibility value attached to it based on its sources' credibility at the time of receiving the information. See also certainty.

defection: When an actor does not fulfil its duties, it is said to defect. In the Pilarcos context, an actor's duties are defined by the eContract; full defection means breaking the contract and not providing compensation. Stretching the contract for personal gains is considered a "partial" defection, which may or may not pass unpunished in an inter-enterprise collaboration. Defection and cooperation are game-theoretical terms; see cooperation for further background.

eCollaboration: A synonym for collaboration as defined in this thesis; the e is used in some of the literature to emphasize this specific type of collaborative activity.

eCommunity: In Pilarcos, eCommunity is used synonymously to a business network.

eContract: Electronic contract, encompassing both a legal agreement and technical governance elements for an inter-enterprise collaboration. In Pilarcos, an eContract is an stateful object controlling the business network. It contains a reference to the business network model that defines the structure of the business network, negotiated service offers fulfilling the roles defined in the network, negotiated configuration for any policies governing the network behaviour, and support for keeping track of the collaboration state, with checks for allowed state transitions.

enterprise: Enterprise is used to refer to any for-profit or non-profit organization which would participate in a collaboration. For example, an enterprise could be a single-person logistics business running a small server process that is able to interface with the Pilarcos collaboration middleware, but implements all management routines itself. At the other size extreme, it could be a multinational software support corporation which uses the Pilarcos middleware to fully automate its collaboration management routines. There are no limitations by enterprise size or domain inherent in the middleware itself, although not all collaborations can be expected to benefit from automated management tools.

epoch: A phase or a period. See reputation epoch and collaboration epoch.

evidence: In TuBE, evidence of an actor's past behaviour takes the form of local observations or external reputation information; the latter is evaluated for credibility. In trust management literature, the term is occasionally used to indicate a special case of irrefutable evidence, such as cryptographically formed, signed and nonrepudiable receipts of successful transactions, which the trustee can store itself and use them to convince the trustor of its good behaviour in the past [Obr04, LSB03].

experience: Experiences about a trustor's past behaviour form the base for its reputation. In TuBE, experiences follow a specific format; see outcome for details. Local experiences are first-hand information based on the output of local monitors, while external experiences are received from reputation networks. When stored in the TuBE reputation data storage, local experiences can be either open, i.e. preliminary and still modifiable, or closed, based on whether the action they are connected to has been fully completed or not. Closed experiences do not technically exist as independent objects, and therefore cannot be changed.

fulfilment: Fulfilment is a TuBE asset used to represent how well the trustor's expectations have been met. While the base of the monetary asset is the wealth of an enterprise, the base of fulfilment is the trend of respected agreements, which reflects on its success. In later publications, this asset has also been referred to as satisfaction, when the literal fulfilment of contracts may need to be complemented with more subjective measures of meeting expectations.

guard: This term is used synonymously to the TuBE trust decision subsystem. The TuBE guard makes trust decisions on whether messages should be allowed through to a computational service. Technically, the guard unit consists of the guard monitor plugin, which is a part of the Pilarcos monitor, and the evaluator class, which also connects to various data sources within TuBE to produce the decision.

impact: The undirected scale of an effect: major, minor or none. See also outcome.

importance: As a factor of a trust decision, importance represents the effect a positive trust decision has, independent of the trustee's behaviour. It represents an a priori incentive to collaborate, which is based on considerations such as not wanting to miss a business opportunity, avoiding compensation required for possibly violating the collaboration contract with a negative decision, or simply gaining positive reputation through cooperation. The factor emulates the business value or business importance of an action.

inter-enterprise collaboration: See collaboration.

interoperability: Interoperability, or the capability to collaborate, is the effective capability to mutually communicate information in order to exchange proposals, requests, results, and commitments (i.e., to exchange speech acts common in business) [KRRM08]. Interoperability can be divided into technical, semantic and pragmatic interoperability. Technical interoperability is concerned with connectivity between the computational services, allowing messages to be transported from one application to another. Semantic interoperability means that the message content becomes understood in the same way by the senders and receivers, both in the sense of information representation and messaging sequences. Pragmatic interoperability captures the willingness of partners to perform the actions needed for the collaboration, both in the sense of capability of and in preferability to the enterprise.

malicious: A malicious actor is highly likely to defect from a collaboration, and may aim to circumvent the trust and reputation management system in order to get away with this. In computer security, malicious actors are traditionally attackers aiming to subvert the targeted system for undefined reasons. In inter-enterprise collaboration, malicious behaviour by rational attackers primarily aims at personal gain at the cost of others, while causing damage to the target is not necessarily a primary goal. The division between cooperative and malicious actors is not black and white in business; instead, the focus is on the incentives for desired and undesired behaviour from the point of view of the designed system.

monetary: A TuBE asset used to represent the financially measurable effects of actions. The monetary asset is based on the wealth of the enterprise.

monitor: A Pilarcos collaboration management service for operational time interoperability checking, breach detection and experience gathering [KMR05].

monitor event: Local experience gathering in TuBE is done by interpreting monitor events into outcomes of actions, based on the local experience analysis policy. For example, a monitor event of the form "sales transaction completed" with a parameter "price" and a value "50" indicates, assuming an appropriate policy, that the action was successfully completed and had a minor positive effect on the monetary and fulfilment assets, and no effect on other assets.

newcomer: A new actor joining a reputation network is referred to as a newcomer. Providing adequate support for newcomers is often the weak spot for reputation and recommender systems, which leads to so-called cold start problems: on one hand, newcomers have trouble gaining a reputation themselves when there is no one around to recommend them, and on the other hand they have to determine what information and what sources in the reputation network itself are reliable and useful to them.

open service ecosystem: The open service ecosystem is the environment where Pilarcos and TuBE operate: the ecosystem consists of the offered services and multiple ongoing collaborations in different phases of their operation, the infrastructure services to support their management, and the repositories of metainformation, such as service offers, business network models and reputation information. The openness refers to the fact that the participants are not fixed or predetermined in any way: a new service provider can enter the ecosystem and collaborate with the actors by publishing a service offer. In addition, new service types and business network models can be defined to change the way the participants collaborate altogether, which means that the ecosystem is constantly evolving. See also: open service market.

open service market: The open service market is essentially a very basic open service ecosystem: it covers the services offered and collaborations taking place, while not indicating the existence (or nonexistence) of a supporting infrastructure. In the TuBE context, this term is used to refer to the market in the general economic sense, while the open service ecosystem is a more specific term for Pilarcos-style environments.

opinion-based reputation system: A category of reputation systems where information is shared as an overall opinion rather than ratings of single transactions [RKK07]. For more information and a comparison of opinion-based and rating-based reputation systems, see rating-based reputation system.

outcome (experience): In TuBE, the completion of an action has an effect on each asset. These effects are represented for each asset on the scale of major negative effect, minor negative effect, no effect, minor positive effect, major positive effect, or unknown effect in cases where the effect cannot be measured. The outcome of an action is the combination of these asset-specific effects.

partner: In this thesis, we use "partner" to refer to actors interacting within a collaboration; it does not refer to long-term business partnerships. A potential partner being evaluated for collaboration becomes the target of a first trust decision. It is generally not yet accepted into the same business network by the trustor, although in some cases a trustor may decide to join a business network and decide only during the operation of the collaboration on whether it will actually allow the other collaborators access to its services.

Pilarcos: Pilarcos is a collection of interconnected infrastructure services for inter-enterprise collaboration management and interoperability, developed by the CINCO group. It can also be described as collaboration management middleware, but the concept of "middleware" should not be interpreted in the narrow sense of a layer between an operating system and applications, but rather as an extension to the enterprise service bus.

policy: A policy is a plan of action guiding decisions; a higher-level policy may consist of a number of more detailed policies. In the TuBE context, the configurable, decision-influencing parts of of the trust management system are referred to as policies, such as the policy for translating monitor events to outcomes, or the policy for how risk tolerance is adjusted according to different importance values. Policies vary between different actors, and are generally referred to as local policies to emphasize this. On the other hand, policies agreed on by collaborating actors are shared through contracts. In contrast, fixed logic designed to be equal across all TuBE instances is not referred to as a policy.

population: In the population process, service offers are matched to roles in a business network model as well as to each other in order to perform static interoperability checking. The product of a population process is a business network proposal containing a draft eContract, which is then negotiated on and refined among the potential participants in the following negotiation process. The population process is typically performed by a populator service.

populator: A populator is an infrastructure service for populating the roles in a business network with service offers. It can be provided by an independent third party or implemented by the business network initiator itself. The populator connects to different repositories to retrieve service offers and service type information, and can be passed partially populated business networks: for example, the initiator calling the populator may have already set its own service offer to fulfil a role in the business network. See also trader for a comparison.

privacy: In Pilarcos, enterprise privacy is defined broadly as the enterprise's ability to control and track information about itself, and information created by or within it.

rating-based reputation system: We divide reputation systems into rating-based and opinion-based systems [RKK07]. In a rating-based system, the reputation information is shared as set of experiences or similar transaction-bound units. In eBay [eBa11], for example, every transaction is rated separately. In an opinion-based system, the reputation may still be based on experiences, but the peers only share their overall gathered opinion of the trustee without reporting single experiences separately; Regret [SS02], EigenTrust [KSGM03] and Unitec [KR03] are examples of such systems. Reputation shared in opinion-based systems cannot always be converted directly into experiences.

reciprocation: The mutual exchange of deeds, such as favour or revenge [MMH02]. Reciprocity norms drive people to return positive and negative actions in kind.

reputation: 1. As a general term, we define reputation as the perception an actor creates through past actions about its intentions and norms [MMH02].

2. As a factor of a trust decision, reputation represents information

about the past actions of an actor. It is a combination of local experiences, gathered through first-hand observations, and external, third-party experiences. Reputation is stored in reputation views.

3. As a TuBE asset, reputation is used to represent the public opinion concerning the enterprise, including its rating in any reputation networks, appearance in the media, and the attitudes of its partners and customers toward it. The reputation asset is based on the good public relations of the enterprise.

reputation epoch: A TuBE reputation epoch denotes a set of experiences from a continuous period of time during which a trustee's behaviour is considered consistent. If the trustee's behaviour changes considerably, a new reputation epoch is started in order to be able to react swiftly to the change on a trust decision policy level, while holding on to the full history of old experiences.

reputation management: Reputation management is the activity of collecting, storing, updating and distributing reputation information on other actors.

reputation network: A reputation network is a source of external reputation information: a network of actors using a reputation system to share their experiences. Typically, reputation sources are networks of peers providing information to each other through various means; however, we include also fully centralized, single-organization information sources, such as credit rating organizations, in our definition. TuBE is designed to be able to connect to multiple reputation networks for gathering information. See also reputation system.

reputation system: A reputation system consists of an information model and a set of algorithms to collect, process and distribute reputation information. A reputation system instance with a given user base and information content is referred to as a reputation network. A distinction is made between the two because the same reputation system, i.e. the core models and algorithms, can be deployed independently in multiple domains, resulting in separate communities acting as reputation sources. The information conversion needed between the reputation network and TuBE depends on the reputation system in use.

reputation update policy: A reputation update policy determines how to integrate new experiences (or entire reputation views) into the existing, locally stored reputation view. In TuBE, the policy for processing local experiences is fixed: once monitor events and other local, trusted information is interpreted into an experience, it is stored without further policy evaluation in the local reputation view. The policy for processing external experiences (or the one for processing full reputation views) can change; the the parameters for the policy are the locally assigned credibility of the new experience, the new experience and the current external reputation view. Further policies control how monitor events and other information are translated to local experiences, and how data from an external reputation system is translated to external experiences in the TuBE format. See also trust decision policy.

reputation view: A reputation view is a data structure that stores experiences in a compressed form that can be quickly processed for making trust decisions. There are two types of reputation views in TuBE: one for storing local experiences, and another for storing external experiences. The local and external reputation views are stored separately and merged for trust decisions only. Their main difference is philosophical: local experiences are more valuable than external experiences, so they are stored in a more stable storage, while an external experience view can be updated to a state that has less (but typically more credible) information. In addition, a local reputation view can temporarily store open experiences, which can be updated while an action is progressing.

risk: As a trust decision factor, risk is the cost-benefit estimate of an action. It is represented as the subjective probabilities of different kinds of outcomes, which affect the trustor's assets. The estimate is built on earlier information represented by reputation. In contrast, the importance factor represents the known, certain outcomes of the action.

risk tolerance: As a trust decision factor, risk tolerance is a policy representing the trustor's willingness to accept the risk involved with the action, both in the sense of finding the probabilities in the risk estimate reasonable and in the sense of considering the risk estimate itself to be composed of sufficient and high-quality information sources. Besides providing a division between acceptable and unacceptable risk, risk tolerance can divide risk estimates into multiple tolerance classes, such as clearly unacceptable, clearly acceptable and gray area. These classes can also be used to provide a subjective, risk-based partial ordering between different trustees or actions, when other factors remain fixed.

role: A role is a part of a business network model, defining the duties of a collaborator in the business network. A role can consist of multiple service types, and can be fulfilled by one business service or by another business network of multiple services, which together provide the service required of the role.

service: A service is a network-accessible object with a published interface and attributes corresponding to one or more service types. A service may consist entirely of software, or it may be a software interface to a physical system. For example, a logistics service can use the standard interface to get the information of what needs to be delivered and where, implement the actual goods transfer with a truck, and deliver a message confirming the delivery electronically. A computational grid service, on the other hand, may be entirely software-based. The current Pilarcos implementation supports Web Services. See also business service.

service ecosystem: See open service ecosystem.

service offer: A service offer advertises a business service that fulfils a particular service type. A service offer consists of a reference to the service type as well as acceptable values to different possible parameters defined in the type, such as quality of service provision or the price of the service.

service provider: A service provider is an enterprise offering a business service. In a legal sense, it is represented as the actor in a contract, but in a technical and trust management sense, the business service acts independently on behalf of, or representing its provider. See also actor.

service type: A formal representation of a specific kind of service. A service type defines the interface of a single service, its inputs and outputs. See also service, role, business network model.

subjective probability: Subjective probability is the probability of an event as estimated based on the information available to the estimator. In the case of TuBE, the estimator is the trustor. Subjective and objective probabilities of positive outcomes are commonly used in trust and reputation modelling within multi-agent systems, particularly with Bayesian reputation models. In contrast to agent-dependent subjective probabilities, objective probability represents the "actual" probability of the event, which in the case of simulated agents is well-defined as it is programmed into the simulator. This kind of internal programming emulates the agent's inherent trustworthiness, which is not as readily measurable in the case of unpredictable human actors.

tolerance: See risk tolerance.

task: In Pilarcos, a task consists of the exchange of one or more messages. While a task and the TuBE concept of action are independent as they serve different purposes, tasks may often prove to be a suitable granularity for trust decisions as well, in which case they can be defined to be equal.

trader: A trader can be seen as a simplified populator that only populates a single role by matching an offer to it, and does no matching between different service offers in the same business network. The trader concept is used in currently existing standards, such as UDDI [BCE$^+$02], and in ODP [ITU97]. In Pilarcos, the trader is replaced by the populator service.

trust: In TuBE, trust is defined as the extent to which an actor is willing to participate in a given action with a given partner, considering the risks and incentives involved. In TuBE, we focus on computational trust used in trust decisions.

trust management: In TuBE, trust management is the activity of upkeeping and processing information which trust decisions are based on.

trustee: The actor target of a trust decision.

trustor: The actor making a trust decision.

trust decision: A trust decision determines whether the trustor considers the involved risks tolerable for a given commitment. A trust decision is based on seven factors: the trustor, trustee and action involved, and the calculated risk, risk tolerance, reputation and importance. The four latter factors are further adjusted by context filters.

trust decision policy: The trust decision policy directs how a trust decision is produced from the information available at decision time. Compare to reputation update policy.

TuBE: TuBE (Trust Based on Evidence) is a trust management system automating routine trust decisions and upkeeping information needed for them. It is a part of the Pilarcos middleware for inter-enterprise collaboration management.

uncertainty: See certainty.

utility (measure): Refers to the utility measurements prevalent in the research on decision-making for rational agents. In this thesis, we use utility synonymously to mean the costs and benefits from an action: negative utility refers to an overall cost, while positive utility means an overall gain. It is originally a game-theoretic term, a measure of satisfaction. Intelligent agents use utility functions to subjectively measure the goodness or badness of specific outcomes, and then aim to make decisions that maximize the values of these functions, i.e. "maximize utility".

web-Pilarcos: Synonym for Pilarcos. For a period of time, web-Pilarcos served as the name of the Web Services based version of the middleware to distinguish it from the old version. The Pilarcos middleware has been previously implemented for e.g. CORBA.

Web Services: The Web Services architecture represents the web-accessible equivalent of Service Oriented Computing (SOA) [BHM+04, Pap03]. The architecture description by W3C defines common elements needed to support interoperability between services. The focus is on standardizing access interfaces to the services, while the architecture sets no limits to the actual implementation of the services or on combining them.

Available online: http://dx.doi.org/10.1007/11429760_6

Appendix II

Sini Ruohomaa and Lea Kutvonen

Trust Management Survey

In Proceedings of the iTrust 3rd International Conference on Trust Management, 23–26, May, 2005, Rocquencourt, France (iTrust 2005), May 2005, pages 77–92.

Available online: http://dx.doi.org/10.1109/ARES.2007.123

Appendix III

Sini Ruohomaa, Lea Kutvonen and Eleni Koutrouli

Reputation Management Survey

In Second International Conference on Availability, Reliability and Security (ARES'07), April 2007, pages 103–111.