Date of acceptance Grade

Instructor

Tool for simulating reputation management algorithms in multiagent systems

Liliya Rudko

Helsinki October 17, 2013

MSc thesis

UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Faculty of Science	Department of Computer Science

Tekijä — Författare — Author
Liliya Rudko

Työn nimi — Arbetets titel — Title

Tool for simulating reputation management algorithms in multiagent systems

Oppiaine — Läroämne — Subject
Computer Science

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
MSc thesis	October 17, 2013	74 pages + 8 appendices

Tiivistelmä — Referat — Abstract

Efficient service-oriented inter-enterprise collaboration focuses on the business processes of the enterprises and hides the technology differences between them. However, such collaboration induces two main challenges. First, there should be an accessible and functioning infrastructure available for the collaboration. Second, as the growing number of participants can lead to the growing level of misbehaving among them, and thus market deterioration, that is why the parties should have common understanding of the behavior that is appropriate and can be trusted.

We focus on the trust relationships between the agents' interactions. Agents make a trust decision before interacting, and this decision among other factors is based on an estimation of another agent's reputation. A reputation management system collects and analyzes interactions experience between agents.

Let us call a person, company or any other possible entity whose goal is to build an infrastructure for the interactions between enterprises using one of the trust and reputation management algorithms as an infrastructure builder. For an infrastructure builder it is important to evaluate and compare reputation management systems, choosing one of them based on the evaluation and comparison results. This remains an open question in research. The thesis aims at supporting the decision-making process of this kind.

We suggest evaluation criteria for a trust or reputation management systems' evaluation. We implement a generic tool which can plug in a trust or reputation management algorithm and simulate the behavior of the multiagent system where every agent follows the same algorithm. We illustrate the tool's support for some of the suggested evaluation criteria. We provide some recommendations for further development of the generic tool for evaluating and comparing the above-mentioned behavior characteristics of different trust and reputation management algorithms.

ACM Computing Classification System (CCS):
Human-centered computing → Collaborative and social computing → Collaborative and social computing systems and tools → Reputation systems
Computing methodologies → Artificial intelligence → Distributed artificial intelligence → Multi-agent systems

Avainsanat — Nyckelord — Keywords
multiagent systems, reputation management systems, reputation management algorithm

Säilytyspaikka — Förvaringsställe — Where deposited
Kumpula science library, serial number C-2013-

Muita tietoja — övriga uppgifter — Additional information

Contents

1 Introduction

Enterprises that provide services differ in terms of business management concepts and technology solutions [KRRM08]. Conventionally, service-oriented inter-enterprise collaboration implied inter-connection of business processes based on homogenized technology. However, effective service-oriented inter-enterprise collaboration should isolate technology differences and focus on business processes. In order to enable the isolation of technology differences, we can for example represent enterprises' services through interfaces of different types and define the ways of inter-connections between these interfaces.

The main advantage of such technology-isolated inter-enterprise collaboration over conventional one is the smaller investments required from the parties for the technology inter-connections. Moreover, the parties can keep their technological autonomy. However, two main challenges arise in this technology-isolated context of inter-enterprise collaboration. First, there should be an accessible and functioning infrastructure available for such collaboration (e.g., to define service interfaces and inter-connections between them) [Ruo12]. Second, as the growing number of participants can lead to the growing level of misbehaving among them, and thus market deterioration, the parties should have common understanding of the behavior that is appropriate and can be trusted.

We represent interactions between enterprises as interactions in a multiagent system. A multiagent system can represent not only network of the enterprises, but also for example a social network. The agents of the system can interact with each other, for example communicate or cooperate. We focus on trust relationships between agents' interactions. We define trust that an agent A puts in an agent B as "willingness of the agent A to take the risk of collaboration with the agent B" [Ruo06]. Agent A is a trustor, agent B is a trustee [KBR05]. Agents make a trust decision before interacting, and this decision can be either binary (e.g., agent either trust or not) or binary with a hierarchy of sub-decisions (e.g., agent trusts given certain conditions) [Ruo06]. Trust decision is based among other factors on estimation of a trustee's reputation. We define reputation as a "perception that an agent creates through past actions about its intentions and norms" [MMH02]. A reputation management system collects and analyzes interactions experience between agents.

There is quite a number of reputation management algorithms suggested for multiagent systems in the research field. The algorithms differ in their approaches to

reputation management, initial assumptions, data models, and technical implementation. Thus evaluating and comparing the algorithms becomes rather challenging.

In this context, Carbo et al. have suggested the Agent Reputation and Trust testbed ART [ART11]; Kerr and Coher have proposed the Trust and Reputation Experimentation and Evaluation Testbed TREET [KeC10]; Chandrasekaran and Esfandiari have suggested the model for a testbed for evaluating reputation systems [ChE11]. ART and TREET simulate basic market place scenarios and thus application specific, whereas the model of Chandrasekaran et al. provides a generic workflow of trust relationships to identify the stages of this workflow that different reputation systems occupy. The model also describes and evaluates certain attack scenarios. An evaluation of characteristics of reputation management systems (e.g., recoverability of an agent's own reputation or an agent's reactivity to changes in behavior of other agents) remains an open question. For an infrastructure builder it is important to evaluate and compare reputation management systems, choosing one of them based on a generic set of evaluation criteria.

The goal of the thesis is to identify the evaluation criteria for trust and reputation management systems, implement a generic tool for evaluating and comparing these systems and illustrate the tool's support for evaluating the algorithms according to the identified criteria.

The research method of the thesis includes a state of the art analysis of the field, design and implementation of a tool for evaluating and comparing reputation management algorithms, representation of one of the reputation management algorithms in the system, design and implementation of benchmark loads based on a set of evaluation criteria, evaluation of the algorithm given the developed loads, and identification of recommendations and needs for future improvements.

The rest of the paper is organized as follows. Chapter 2 presents environment of the problem, the role of reputation management in making trust decisions, and reputation management concepts and terminology used in the thesis. Chapter 3 gives an overview of the state of the art of reputation management algorithms and existing models of testbeds for the comparison of reputation management algorithms. Chapter 4 describes the evaluated system's behavior characteristics and presents our tool architecture and interface. Chapter 5 presents the chosen algorithm for implementation in the tool and its representation in the system. Chapter 6 demonstrates evaluation examples of the chosen reputation management algorithm. In Chapter 7 we conclude our research results and identify future work.

2 Trust and reputation management concepts

Different enterprises specialize in various professional fields (e.g., railway transportations or translations between different languages). The expertise and field coverage of one enterprise may be not enough for offering a complete product or providing turn key service to its customers. For that reason enterprises start to collaborate with each other. Collaboration reveals possibilities for the enterprises to focus on their own expertise and at the same time provide a complete product, wide range of services or turn key services to their customers. Moreover, inter-enterprise collaboration opens new possibilities for small and medium sized companies to compete with the large corporations in the same professional field [RuK08].

For example we can consider a company X that provides the state railway transportation. This company has a simple web-site which contains the most important information, including for example timetable and prices. However, passengers cannot buy tickets using this web-site and have to queue in the X's offices. X can provide wider range of services to its customers if it collaborates with the company that makes the integration of the payment providers with the X's web-site and X's information system.

Conventionally, inter-enterprise collaboration has implied integration of business processes based on the compatible technology. Making technology compatible usually implicates an expensive and time-consuming process of the integration of different information systems of the enterprises.

To make conventional inter-enterprise collaboration more effective, enterprises should isolate technology differences and focus on their business processes [KRR08]. In order to enable the isolation of technology differences, we can for example represent enterprises' services through interfaces of different types and define the ways of interconnections between these interfaces.

For example in the collaboration between the above-presented state company that provides railway transportation and the government, technology isolation could mean the following. The government has a web-portal for the organizations in which the government has its share. For the state railway company the government can specify the amount of subsidies or required directions of the railways extension in that web-portal. The information system of the railway company automatically connects to the web portal, extracts required information and presents it to the headquarter of the railway company. At the same time the government web-portal

automatically connects to the information system of the railway company to extract weekly/monthly/yearly reports and presents them to the government officials.

The main advantage of such technology-isolated inter-enterprise collaboration over a conventional one is that the parties can keep their technological autonomy. Furthermore, such collaboration allows to increase the number of possible participants either within one deal or total number of participants interacting with each other.

However, there are two main challenges connected with such technology-isolated inter-enterprise collaboration. First, an accessible and functioning infrastructure should be available to enable the collaboration (e.g., to define service interfaces and inter-connections between them) [Ruo12]. Second, as collaboration in this new environment is much easier and leads to new possible interactions, the trust between new partners cannot be formed in the conventional way when trust relationships are built during a long time period and common experience. In other words, a growing number of participants or possible partners can lead to a growing level of misbehavior among them, and thus market deterioration.

Thereby the collaborating parties should have common understanding of the behavior that is appropriate and can be trusted. In the conditions of easy collaboration between market participants there is a need for a system that helps them to identify whom to trust, encourage trustworthy behavior and discourage those participants who are dishonest [RZF00].

2.1 Risk-aware trust decisions

This section presents the main terms within the thesis. We also describe the steps of a trust decision process in a multiagent system and specify the place of reputation management in this process.

In the thesis we represent interactions between enterprises as interactions in an **open multiagent system**. A multiagent system implies that agents behave in an autonomous manner in order to achieve their own goals, while open multiagent system implies that agents can freely join or leave the system at any time [HJS06]. A multiagent system can represent not only a network of enterprises, but also for example a network of human beings - social network.

The agents of a multiagent system can interact with each other (e.g., communicate or collaborate). We focus on trust relationships between agents. **Trust** that a trustor puts in a trustee is an "extent to which a trustor is willing to participate in a given

action with a trustee, considering the risks and incentives involved" [Ruo12, KBR05]. Figure 2.1 represents a schematic view of a trustor that places its trust in a trustee.

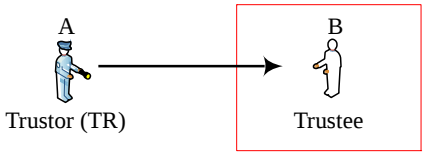
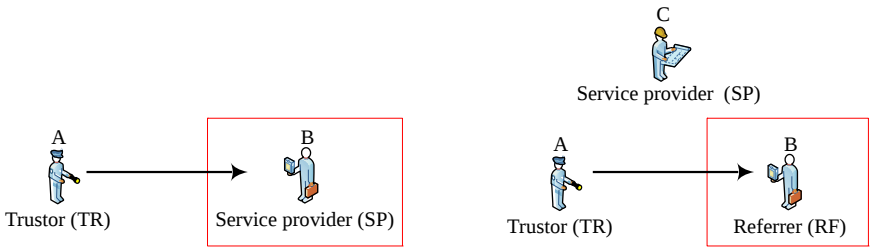


Figure 2.1: A trustor and a trustee.

In our multiagent system an instance of interaction between agents is a service. We define the following roles of the agents: a service provider SP, referrer RF (to a service provider) and a trustor TR. Both a service provider and a referrer can be trustees for a trustor. Any agent in the system can perform one of the roles at a time. For example an agent A can be a trustor in one interaction and identify its trust placed in an agent B who is a service provider in this interaction. Figure 2.2(a) reflects this case. However, concerning different interaction A can still perform the role of a trustor, but view B as a referrer to other service providers. Figure 2.2(b) reflects this case.



(a) A trustor and a service provider as a trustee.

(b) A trustor and a referrer as a trustee.

Figure 2.2: Different roles of a trustee.

Agents make **trust decisions** based on evaluation and comparison of involved risks and benefits of an interaction [Ruo12]. Trust decisions are made before actual interaction, or during this interaction if any risk-relevant actions are involved [RuK08]. Our view of information model for a process of taking trust decisions is based on the model presented in the Trust Based on Evidence (TuBE) system [Ruo12], which includes the following decision factors: reputation, risk, importance, risk tolerance and context (Figure 2.3).

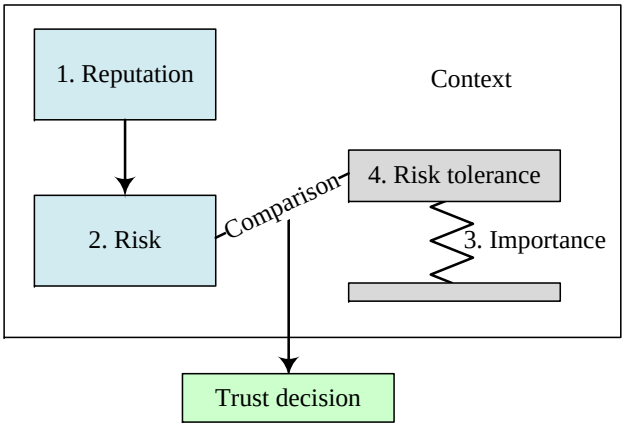


Figure 2.3: Information model for a trust decision process.

Trust decision is based on the comparison between estimated risk involved in the action and level of risk tolerance identified for the action. Risk estimation is based on the trustee's reputation. Risk tolerance is based on the business importance of the action. Context takes into account any fluctuations in the environment and local policy within a trustor enterprise, adjusting the aforementioned decision factors according to these changes.

Any trust decision and its underlying risk analysis are aimed to protect or fulfill interests of a trustor in certain areas. Upon the collaboration experience trustor evaluates the trustee in these areas in order to store the information for future trust decisions with the same trustee concerning the same service type. Different trust management systems define the way of estimating a trustee concerning particular service type differently. Some systems (e.g. EigenTrust [KSG03] or PeerTrust [XiL04]) evaluate collaboration experience in a single dimension (e.g., experience or money gain/loss), others (e.g. Pinnocchio [FKO04], FIRE [HJS06] or the TuBE [Ruo12]) - by a set of dimensions that can represent different collaboration aspects (e.g., financial, security and contract satisfaction). Mostly in the latter systems a different set of dimensions can be assigned also to different service types.

Pinocchio [FKO04] for example allows defining different dimensions of service quality for different service types (e.g., quality of final product and profitability of service). FIRE [HJS06] allows rating service providers from the point of view of different dimensions, however not distinguishing between different service types.

The TuBE system [Ruo12] defines monetary, reputation, control, and satisfaction as guarded assets. The system analyzes potential outcome or changes for each of

these assets for the trustor. The monetary asset represents monetary gains or losses connected with any artifacts that can have monetary value (e.g., cost of service or a good, compensations, or gained profit). The reputation asset represents a trustor's reputation in the community both as a service provider and as a provider of recommendations about other market participants (e.g., how a collaboration will affect these reputation views of the trustor in the community). The control asset represents security and privacy issues (e.g., information or physical security), autonomy and independence, reliability and availability. The satisfaction asset represents the accomplishment of the trustor's expectations about the trustee's behavior (e.g., quality of the service received, prompt response or the level of reality matching with expectations).

In general terms we can represent dimensions of the collaboration aspects in the following form. Let a trust management system to trace a set of collaboration dimensions D, $D = \{dimension_0, dimension_1, ..., dimension_N\}$. Indexes of the dimensions range from 0 to $|D| - 1$. In every dimension there is a set of possible collaboration outcomes O, $O = \{outcome_0, outcome_1, ..., outcome_M\}$ that can be represented as a set of integers (e.g., for the *experience* dimension $outcome_0 = 0$ is unknown effect, $outcome_1 = 1$ is positive effect, $outcome_2 = 2$ is negative effect). Indexes of the outcomes range from 0 to $|O| - 1$. In general terms, for every dimension the *unknown effect* or similar outcome that contains the number of experiences where no outcome type can be determined should be tracked in order to evaluate the quality of available information [RuK08].

Thus, the process of taking a trust decision in the given environment flows in two directions - one of them aims at identifying risk associated with the action, while another one aims at identifying risk tolerance accepted for the action. The thesis focuses on the reputation factor in the whole chain of taking trust decisions, as reputation is a key input to these decisions. However, we can not eliminate risk analysis and its comparison with the risk tolerance either, as the reputation view is based on previous experience, but in order to obtain this experience, we need to take trust decisions of whether collaborate or not with a certain trustee.

Further we present a generic view of the important aspects within two main tracks of a trust decision process: risk estimation based on reputation information and risk tolerance identification based on importance of the action [RuK08]. However, more attention is paid to the reputation management in the whole chain of taking trust decisions. Context adjustment is out of scope of the current work.

2.2 Reputation management concepts

This section represents our notion of reputation. We describe possible types of reputation and their representation. We unify the types of reputation and their representation among the algorithms to make them comparable between each other, however this unification is generic enough to embrace specific visions of different algorithms. Moreover, we identify how reputation can be represented in our system. The section helps to map reputation representation of a reputation management algorithm to the generic reputation representation of the system, which is crucial for further plugging in the reputation management algorithm into the system.

2.2.1 Types of reputation

Reputation is a "perception that an agent creates through past actions about its intentions and norms" [MMH02]. Referral is the information about agents' reputation. A source of a referral is a referrer. Referrals can be either direct or indirect. When a referrer has a first-hand experience with the service provider, their referrals are direct. When a referrer has gone through mediators before reaching the trustor, their referrals are indirect [Ruo12].

A **reputation management system** collects, analyzes and distributes information about interaction experiences between agents [RKK07]. Reputation management system with the agents of the system form a reputation network [Ruo12].

An agent's reputation can be either **global**, which represents a shared reputation value for all the system members, or **subjective**, which depends on the trustor that evaluates it. Subjective reputation is based on aggregation of the direct local experiences with the experiences of other agents. From the point of view of a trustor who stores experience information, local direct experiences form **local** reputation, while aggregation of the third-party experiences forms **external** reputation. Figure 2.4 depicts these types of reputation.

As global and external reputation values are just different ways of representing the reputation view from the outside the given trustor system, they are usually mutually exclusive. Reputation algorithms identify local and either global or external reputation values. Our model originally contains local, external and subjective reputation representations, although global reputation can be mapped into the model instead of the external one. Generally speaking, our system allows reputation representation consisting of up to three different values, one of which can be based on two others.

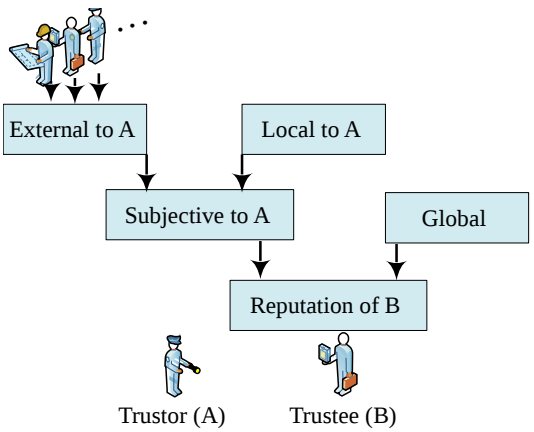


Figure 2.4: Types of reputation.

Local, global and external reputation values are stored separately in our system, as in some scenarios of decision taking these values can be used separately. In the general scenario these values are aggregated only at the moment of decision taking. Below we consider possible behavior scenarios of an agent where local, global and external reputation values can be used separately.

If an agent is new to a network, it has no direct experience with any of the service providers, which means that it has no local reputation for any of them yet. In this case it may use either the global reputation, if it is available, or external reputation of the requested service provider. Global reputation of a service provider is the same from the point of view of all the agents, that is why it is sufficient to contact one of them or the central system. However, external reputation can be obtained based on one, some or all the agents experiences in the network, which will result in different reputation values. The approach is specified by the chosen reputation management algorithm. Thus in this scenario agent can separately use either global or external reputation of the service provider.

If an agent is not new to the network, it has its own local reputation information. In case it cannot obtain global or external reputation for some reasons, it can simply rely on its own local information. Thus in this scenario local reputation can be used separately.

In a common case an agent has its own local reputation information, it requires either global reputation or external reputation from the third-party experiences. Then it aggregates these types of information into a complete picture of service providers and finally chooses one of them to interact with based on this information.

2.2.2 Reputation representation

This section shows the representation of local and external reputation views, and some implementation solutions for the reputation representation in our system. We suppose that local reputation view contains local type of reputation, while external reputation view contains external and global types of reputation. Reputation representation of local and external reputation views of a trustee concerning a particular service from a trustor's point of view follow the same format. Reputation representation contains **number of experiences** of every outcome for every dimension specified for the provided service. Alongside with this representation, it is important for every reputation view to assign the overall **credibility** value that estimates how accurate and useful this reputation information appears to the trustor [Ruo12].

Further we take a closer look at the most common problem connected with gathering the information about number of experiences with a particular service provider concerning a particular service. Moreover, we shed more light on the notion of credibility of the reputation value. Finally, we present some implementation solutions taken for the reputation representation in our system.

Commonly in the trust management systems reputation information represents experience outcomes regardless of the actions that led to these outcomes. For example there is a service provider that sells books. Service type in this case can be *selling books*. The service provider can sell both new and second-hand books. Suppose there is one dimension that we are interested in for this service type — *experience* — that can have the following outcomes: 0 as *unknown effect*, 1 as *positive experience*, and 2 as *negative experience*.

As we do not store actions that led to the outcomes, our reputation information will contain the number of experiences with this service provider concerning selling books: 10 positive and 3 negative experiences. Based on this we take our future decisions of collaborating or not with the same service provider. However, this reputation information connected to actions would be represented in the following way: selling new books resulted in 10 positive and 0 negative experiences, while selling second-hand books resulted in 3 negative and 0 positive experience. Based on this information the trustor would make more accurate predictions about the service provider behavior in the particular action. We believe that if it is important for a trustor to connect actions with their outcomes, it is possible to detail service types to the required level. Thus instead of having one service type *selling books*, we could identify two of them: *selling new books* and *selling second-hand books*. So,

every important action can be traced by introducing a separate service type.

However, in this case the **sparsity** of information can become a potential challenge. A multiagent system contains quite a number of agents and services. However, even very active agents will rate just a small portion of all the available services, while even the most popular services will have a small portion of ratings from the agents with respect to the overall number of the ratings in the system [PPK05]. Splitting one service type to several increases the number of available types of services in a multiagent system and thus increases the sparsity of information. Information sparsity becomes a challenge for an algorithm to gather the experience information, as it becomes a problem to locate the sources of this information.

Knot-aware trust based reputation model [GGH08] for example suggests one possible solution to the information sparsity problem. The model identifies the particular group of similar members inside the entire multiagent system and specifies an agent's behavior in such a way that the agent addresses to the members of its group in the first place.

The overall **credibility** for a reputation value estimates how accurate and useful this reputation information appears to the trustor [Ruo12]. Credibility value can be for example a real number in [0..1]. In this case we suppose that credibility of the local reputation view equals to 1, as the trustor believes in its own first-hand experience. In general terms, if there are several communities, each of which uses different trust or reputation management system, a trustor has its representative agents in every community. Credibility of every piece of external information is a combination of a credibility of the source agent with the credibility of the source agent's community. Overall credibility of the external reputation information is based on the combination of the credibilities of its different pieces.

In our model we simulate a multiagent system as a single community, every agent of which uses the same trust and reputation management system. A credibility of the external piece of information in this case contains a source agent's credibility only. Thus, we omit credibility value for the whole community.

Different reputation management algorithms specify differently the algorithms for combining different reputation values between each other or with the trustor's local information, as well as algorithms for combining credibility of the gathered information. Section 3.1 presents an overview of the main differences in reputation management algorithms, as well as differences in their assumptions.

Further we present some implementation solutions taken for the reputation representation in our system. Reputation.java class is responsible for representing reputation. This class should be generic enough to be able to represent possible reputation view of different reputation management algorithms. Particular reputation representation is defined by the reputation management algorithm used in a multiagent system. Table 2.1 describes the fields of the class.

Table 2.1: Fields of Reputation.class in the system.

Field type	Field name	Field description
int	agentId	Id of the agent whose reputation value this object represents
Pair	localReputation	Local reputation value of a trustee. Field contains a pair of objects. First object is an object of local reputation, while second is an object of a credibility value of the local reputation view.
Pair	externalReputation	External or global reputation value of a trustee. First object of a pair is an object of external reputation. Second object in a pair is an object of credibility value for the external reputation view.
Pair	subjectiveReputation	Subjective reputation value of a trustee. First object of a pair is an object of subjective reputation. Second object in a pair is an object of credibility value for the subjective reputation view.
long	timestamp	Time at which this reputation object was created

Reputation representation is important because it is the main calculation unit of any reputation management algorithm. For this reason some reputation management algorithms consider up to three different reputation views to compute the single reputation value at a time. Different reputation views allow to make calculations of the overall reputation value more accurate. However, it may become a challenging mathematical task to combine them and proof the validity of the suggested solution.

Moreover, reputation representation also determines how reputation information can be shared in the system. However the main goal of reputation representation is using it in trust decisions. Section 2.3 elaborates how reputation information can used in a trust decision.

2.3 Using reputation information in a trust decision

The section evolves Section 2.1 given our knowledge of Section 2.2. It shows theoretical representation of every step in a trust decision process, including representation of reputation and reputation management in this process.

Our general model of risk analysis based on reputation information is similar to one presented in the TuBE system [RuK08]. Reputation U of a trustee consists of two halves: local reputation U^{local} and external reputation U^{ext}, combined from multiple sources.

Each of these reputations consists of vectors that contain a number of experiences for all the possible outcomes in O for all the dimensions in D, $d = 0, 1, ..., |D| - 1$. In other words, there are $|D|$ vectors, each of which corresponds to the dimension d and contains the counters of all the experienced outcomes in O. Based on this information we can identify the total number of experiences with a trustee. Let E^{local} and E^{ext} be the whole set of local and external experiences respectively. Then the number of local and external experiences with the trustee respectively are $|E^{local}|$ and $|E^{ext}|$.

In this generic model we assume that there is an outcome *unknown effect* in O. Calculating the number of these outcomes separately allows us to identify the proportion of uncertain values. Thus, q_d^{local} and q_d^{ext} correspond to the number of experiences in $|E^{local}|$ and $|E^{ext}|$ accordingly where a trustor registered *unknown effect* outcome in the dimension d. Then, proportion of uncertain values for local and external experiences in the dimension d would be $\frac{q_d^{local}}{|E_{local}|}$ and $\frac{q_d^{ext}}{|E_{ext}|}$ respectively.

Local and external reputation views are represented also with the credibility scores c^{local} or c^{ext} respectively. The local credibility value c^{local} is the maximum value in the range of credibility values (e.g., in case credibility lies in [0..1], $c^{local} = 1$), which indicates that a trustor believes in its own first-hand experience. The external credibility value c^{ext} is the combined credibility value of all the third-party experiences.

Before calculating the risk associated with the action, we need to combine local and external reputations in the overall reputation view U. The combination includes: 1) calculation of the overall number of experiences, $|E| = |E^{local}| + |E^{ext}|$; 2) calculation of the overall number of experiences with *unknown effect* outcome, $q_d = q_d^{local} + q_d^{ext}$; and 3) aggregation of the credibility values c^{local} and c^{ext} in the overall credibility value c. The details and scope of the reputation representations as well as their

special features in the merging process are defined by the concrete reputation management algorithm used in the trust management system. Above we have outlined the generic framework for this process.

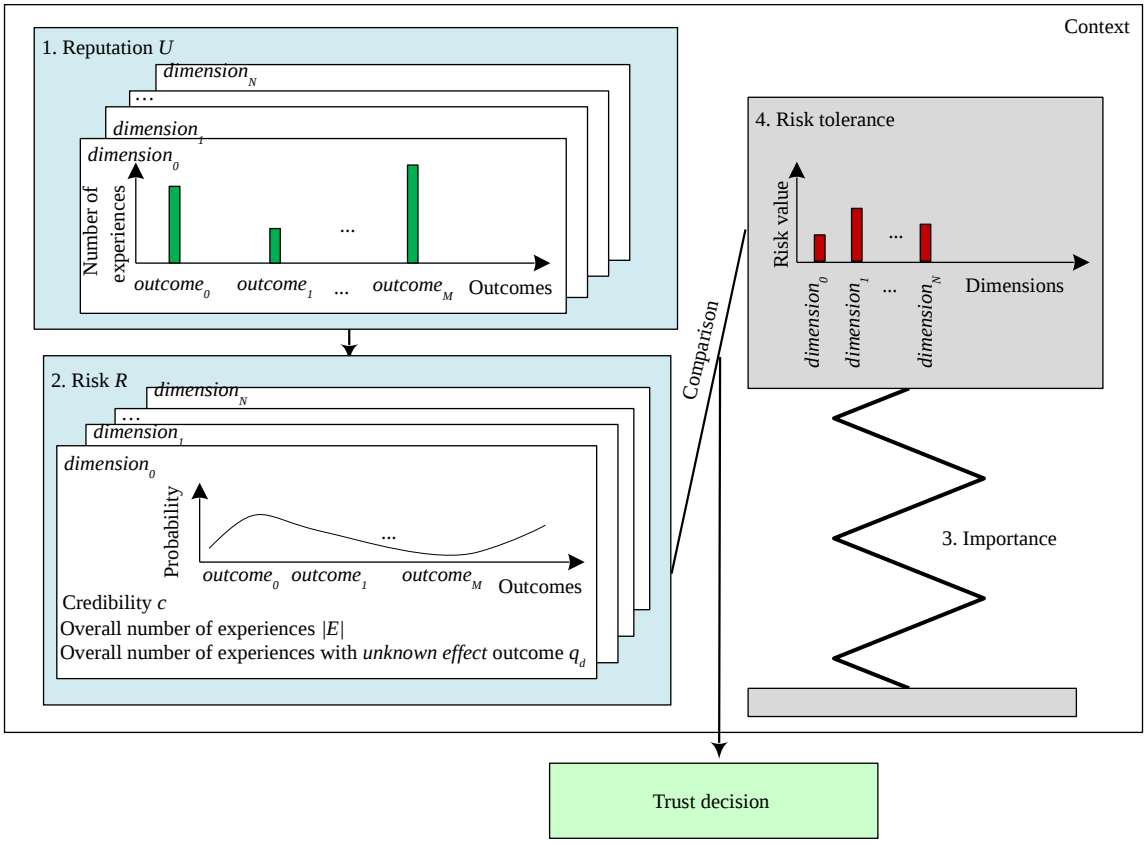


Figure 2.5: Information models of trust decision process.

Risk R of an action contains $|D|$ vectors $\boldsymbol{r_d}$, $d = 0, 1, ..., |D| - 1$, one for each dimension. Thus, $R = (\boldsymbol{r_0}, \boldsymbol{r_1}, ..., \boldsymbol{r_{|D|-1}})$. Every vector $\boldsymbol{r_d}$ in general terms can be represented as $\boldsymbol{r_d} = (\boldsymbol{p_d}, |E|, \mathrm{c}, \mathrm{q}_d)$. In this formula $\boldsymbol{p_d}$ is the vector of probabilities for every possible outcome O in the dimension d, excluding *unknown effect* outcomes, as they are separately represented by q_d. The overall proportion of uncertain values can be identified by the following formula $\frac{q_d}{|E|}$ [RuK08].

In the process of taking trust decisions, presented in Figure 2.3, after we have identified risk based on the reputation information and associated with the action, we define the level of risk tolerance based on the importance of the action. Importance of an action and level of risk tolerance do not depend on a trustee's previous behavior, they are mostly connected with the particular action between the certain trustor and trustee [RuK08]. The more important action the higher risk tolerance associated

with it. Risk tolerance can be represented as a vector $T = (t_0, t_1, ..., t_{|D|-1})$, containing $|D|$ values for each of the dimensions that represent the minimal acceptable ("bottom" limit) for the risk values in the risk vectors $\boldsymbol{r_d}$.

When risk analysis is completed and risk tolerance vector is identified, the comparison between these vectors follows. Figure 2.5 enhances Figure 2.3, representing important information models for each step in a trust decision taking process. In case every risk value in the risk vectors $\boldsymbol{r_d}$ stays within the limited values of the risk tolerance vector, a trust decision is positive, otherwise it is negative.

3 State of the art

This section represents the state of the art in the field of reputation management systems. The aim of the section is to present the design features of the general reputation and trust management flow supposed in our system, based on the analysis of current reputation management algorithms. Moreover, we aim to reveal the place of our system among the current systems whose goal is to evaluate and compare different reputation management algorithms.

To serve our goal, first we present classifications of reputation management algorithms and expose the place of our general reputation management algorithm in these classifications. Second, we have a closer look at distributed reputation management algorithms, as our system is meant for evaluation and comparison of such algorithms. We present some design features of our general reputation management algorithm based on the analysis of the distributed reputation management algorithms. Finally, we identify the place of our system among the systems with the same goal of evaluating and comparing different reputation management systems.

3.1 Classification of reputation management algorithms

We present two different classifications of reputation management algorithms. One of them is based on the technical implementation of the algorithms and distinguishes centralized and distributed algorithms. Another classification is based on the information flows in the algorithms and distinguishes different data sending patterns and data sharing modes of the algorithms.

Figure 3.1 presents classification of reputation management systems based on their technical implementation. In a **centralized** reputation management system all the members have an access to the shared information, which is usually maintained by the central server. The main purpose of a centralized reputation or trust management system is to allow easy collaboration between the existing members of the network. Thus, in case a centralized reputation or trust management system becomes successfully deployed, more and more participants are joining the network and the system becomes globally accessible. Centralized reputation management system are used for example in such on-line market systems as eBay and Amazon.

At some point if a centralized entity does not scale itself or is not adjusted in the required way to serve the increasing number of the network members, it becomes

difficult for the system to govern all the members. The huge number of members implies also a growing number of misbehaving among them. Theoretically the number of members in the network can be unlimited. However practically a centralized entity has some limits of scaling according to the limited resources around the world. Thus, at some point the system's performance deterioration or even crashing becomes inevitable.

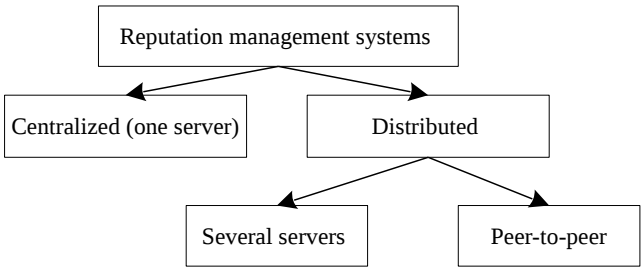


Figure 3.1: Classification of reputation management systems.

Decentralized or **distributed** reputation systems become an alternative to centralized ones, providing new ways of reputation and trust management systems deployment. Thus for example distributed trust systems provide possibilities for incorporating various reputation management algorithms between the network members.

A pure centralized reputation management algorithm implies that a multiagent system has one server, which obtains the data from the agents and performs the calculations. A distributed reputation management algorithm implies that either a multiagent system does not have a central server altogether, or it has several servers that execute the algorithm and maintain consistency of the data between each other. In the former case every agent implements the reputation management algorithm locally and shares the data according to a defined protocol.

Our model focuses on distributed peer-to-peer reputation management algorithms, as they are quite numerous compared to other ones, and it has become hard to compare these algorithms between each other [KeC10]. However, both distributed algorithms with several servers and centralized algorithms can be mapped to our model as well. In case of centralized reputation management algorithms, we can consider them as distributed reputation management algorithms with only one provider of recommendations. Similarly for the distributed reputation management algorithms with several servers we can consider a distributed reputation management algorithm with several providers of recommendations.

The second classification is based on the information flows of the reputation management systems. Chadwick [Cha05] distinguishes **data sending patterns** and **data sharing modes** that can be used in any of the systems' types presented in Figure 3.1. Data sending patterns include raw data and reputation values. Data sharing modes include push and pull mode. Figure 3.2 depicts data sending patterns and data sharing modes.

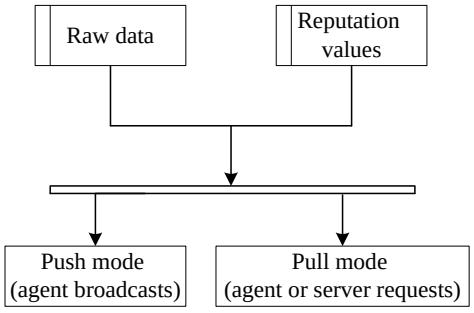


Figure 3.2: Data sending patterns and modes.

In any of the specified type of reputation management system data can be obtained either in a raw format or as a computed reputation values. A centralized server can obtain raw data, perform calculations of reputation values and provide the results. Similarly, centralized server can obtain calculated reputation values from the agents, collate them and provide the results. In the latter case Chadwick also supposes that agents can calculate reputation values based on their own reputation management algorithms that can be different between the agents [Cha05]. However, we assume that it is better for the agents to use the same reputation management algorithm, as these algorithms have different assumptions, areas of application and protocols. Collating reputation values that were computed using different logic can provide unreliable and inconsistent results to be used by other agents in their reputation calculations. However agents can still use different decision policies locally that does not affect any results of reputation calculation of other agents.

In the same fashion, agents can provide either raw data or already computed reputation values to one of the servers in a distributed reputation management system with several servers. Similarly, depending on the distributed algorithm, agents can obtain either raw data or computed reputation values from other agents in a distributed peer-to-peer reputation management system.

Raw data or reputation values can be obtained by one of the two different data sharing modes: push or pull. Push mode implies that the agents spontaneously

(e.g., after performing the transaction) provide the data to the network. If there is one central or several distributed servers in a multiagent system, agents provide the data either to central or one of the distributed servers (e.g., the closest one). In the push mode for the peer-to-peer distributed reputation management systems agents can broadcast the data to all the other agents or to the agents specified by the reputation management algorithm.

Pull mode implies that either a server or an agent in need requests the data from other agents. For all the algorithms there is a challenge of finding or choosing the agents to gather the data from. For the algorithms with either central or distributed servers there arises also another issue of specifying the time intervals or patterns of pulling the data.

According to Chadwick, the data sending pattern which is mostly protected from being misused is sending computed reputation values [Cha05]. In this case it is more difficult for a server or another agent to skew the results, as it does not have access to all the original information. In terms of the data sharing mode, Chadwick distinguishes pull mode as the most reliable. However, there should be a publicly available list of all the agents of the network, which should be protected against unauthorized modifications. Moreover, there should be the possibility of multiple registering for the same agent. Given these conditions pull mode can be repeated and that is why is more transparent.

Our general model of reputation management systems is based on the distributed peer-to-peer approach. The data sending pattern can be either raw data or reputation values and is specified by a particular reputation management algorithm. The default data sending mode is pull, where agent requests data from all the other agents when it needs the information.

3.2 Distributed reputation management algorithms

This section presents an overview of existing distributed reputation management algorithms. Moreover, we identify possible vulnerabilities of distributed reputation management systems. The section forms a basis for our system to be able to plug in different reputation management algorithms with their special features. As well as the knowledge of possible vulnerabilities of reputation management algorithms helps us to identify the ways of comparing the algorithms in terms of their resistance to different vulnerabilities.

A distributed reputation management algorithm can either have distributed stores to where trustors submit their experience information, or every trustor can keep its information locally and provide it when other agents request [JIB07]. A trustor when searching for the external information can either refer to one of the distributed stores or try to contact other agents. In the latter case the number of agents and their selection pattern is specified by a reputation management algorithm. After obtaining external information, the trustor aggregates it with the local information if it is available. The aggregation manner is specified by a reputation management algorithm, which can also take into account the credibility of the sources.

These steps lead to a complete view from a trustor's perspective on a requested service. The trustor can either specify a service provider while requesting the information about the service, or it can get the information about all the service providers that other agents had experience with concerning the requested service. The trustor eventually chooses one of the service providers to collaborate with.

Above we have described a general scenario for a reputation management system. Different reputation management systems can omit or change different parts of the scenario. Some systems for example may not consider local reputation at all. Others may first aggregate external information into the total external view and then combine it with the local information, or aggregate every piece of external information with the local and then combine those values together.

Reputation management algorithms differ in two main aspects [JIB07]: distributed communication protocol and reputation calculation method. Distributed communication protocol can specify for example general assumptions for the reputation management algorithm, information sources, the way in which experience information is stored (locally or in the distributed stores) and the pattern of choosing agents or stores for obtaining the information. Reputation calculation method is used by a trustor locally to compute the reputation view of a trustee.

Further we take a closer look on these aspects. We illustrate dimensions for the distribution communication protocol, and how they are covered in our system. We also review different existing reputation and trust calculation methods.

3.2.1 Distributed communication protocol

Below we consider four different dimensions of a distributed communication protocol: information source, visibility type of a trustee's reputation, model's granularity

and agents' behavior assumptions. Moreover, we outline the design assumptions of distributed communication protocol used in our system, main purpose of which is to evaluate and compare distributed reputation management algorithms.

The first dimension of a distributed communication protocol is an **information source**. Sabater et. al distinguish the following information sources in a reputation management algorithm: direct experience, witness information, sociological information and prejudice [SaS05].

Direct experience includes either information about direct interactions of a trustor with a service provider or local observation of other agents' experiences. Observation information can be quite inaccurate and contain a certain level of noise, that is why it is usually not used by reputation management algorithms [SaS05].

Witness information is the information obtained from other members of the community. This information can include either their local information (e.g., observations and experiences) or the information that was obtained by them from other agents.

Social relationships between agents are a source of sociological information [SaS05]. Information about social relationships can include for example information about different roles that the agents can play in a network or their characterization by different social factors (e.g., competition and dependence).

Prejudice information is based on the analysis of a trustor's remarkable features (e.g., brand name and a certain behavior), which allows a trustor to consider a trustee to be part of some community and which influences the reputation value given to the trustee [SaS05].

In our system we include information about direct interactions and witness information about direct experiences of other members. Thus, we do not include both local and witness observation information, as it can be quite inaccurate and is not included in most of the reputation management algorithms [SaS05]. Moreover, we do not include sociological and prejudice information, as these types of information sources are also seldom considered in the reputation management algorithms. However in case they are considered, reputation management algorithm becomes rather complicated, which also leads to complexity of the agents' structure and behavior in the network from the system's implementation point of view [SaS05].

The second dimension Sabater et. al consider is a **visibility type of a trustee's reputation**. They distinguish global and individual visibility types, which is similar to the classification presented in Figure 2.4 [SaS05].

Global visibility implies that all the agents of the system share the same reputation value for any other agent. In most cases it is achieved by a central storage, where the information is updated after every interaction between agents. Every trustor that requires the reputation information of a service provider contacts this storage. However, a global reputation view can also be managed without a central storage. In this case a problem of maintaining the consistency of the data arises. For this reason, global reputation view is mostly used in centralized reputation management algorithms or in distributed reputation management algorithms that are meant for small or medium sized communities [SaS05].

Individual reputation visibility implies that every agent computes the reputation value of a trustee locally. This reputation view can contain local direct experiences and witness experiences. Reputation management algorithms mainly use the individual visibility type [SaS05].

Our system implements individual reputation visibility as described in Chapter 2. However, some distributed reputation management algorithms calculate both global and local reputation values (e.g., knot-aware trust-based reputation model [GGH08]). Our system supports this representation as well, viewing external reputation as global. Figure 2.4 depicts the types of reputation considered in the system.

The next dimension of a distributed communication protocol is **model's granularity**. Sabater et. al distinguish single-context and multi-context reputation or trust management systems [SaS05].

As a service provider can offer different services in different areas (e.g., selling sportswear and selling beverages), a trustor can assign a general reputation value to the service provider without distinguishing between their services, or it can assign different reputation values, one for each service. In the former case we deal with a single-context reputation value, in the latter with a multi-context [SaS05].

Introducing multi-context values by simply using single-context values for different services causes information scarcity problem that was discussed in Chapter 2 [SaS05]. Sabater et. al suggest that multi-context values should provide a different structure of representing various reputation values assigned to different services of the same service provider [SaS05].

In our system we implement multi-context granularity, which can be also simplified and used as a single-context. We identify service domains, service types within the domain and service dimensions. Service domain groups the services that belong

to the same area (e.g., books domain and web-sites domain). Service types are services itself in the specified service domains (e.g., selling books, renewing books and providing books for reading). Service dimensions are dimensions for rating a service type (e.g., quality of a service, timeliness and quality of assistance) that can be assigned differently for different service types. Agents can provide various service types to each other.

The final dimension for a distributed communication protocol is **agents' behavior assumptions**. Sabater et. al distinguish the following assumptions of agents' cheating behavior: agents are always honest; agents can hide or modify the information, but they do not lie; and agents can lie [SaS05].

These agent's behavior assumptions are specific to a referrer role of an agent. Currently our system does not support evaluation of the referrer's behavior, however as a part of a future work there can be specified the patterns for the referrers' behavior, which can represent different cheating behavior patterns, including the ones indicated by Sabater et. al.

3.2.2 Reputation calculation methods

Below we present different reputation calculation methods [JIB07] for our model to consider their special features and make possible to embed them to our system. If we deal with a reputation management algorithm only, trust calculation methods are usually not considered there. However, most commonly reputation management algorithm is embedded into a trust management system. In this case calculation methods of reputation and trust go hand in hand, and it may be quite difficult to separate them.

One calculation method can be used by some authors to identify reputation values, while the same calculation method can be used by other authors to identify trust. This mainly depends on the definition of reputation and trust assumed by the authors. For this reason, some authors can also calculate both reputation and trust by the same method, as the notion of reputation and trust are interchangeable in their definition.

First we focus on the reputation calculation methods, specifying which of them are also used for the trust calculation. Second we present general trust calculation methods that can be applied to any reputation management algorithm in case it does not specify any of the trust calculation methods.

Further we present the following reputation calculation methods: basic calculations, discrete models, probabilistic models, probability distribution models, belief models, fuzzy logic models, and flow models. Also we present the threshold and ranking methods that can be used for making trust decisions.

Basic calculations are the simplest form of an agent's reputation calculation. One method of the basic calculations implies summation of positive and subtraction of negative experiences with an agent. This calculation method was used for example in the reputation forum of eBay according to 2002 research results [ReZ02].

Another method represents reputation as an average value of all the experiences with an agent. This method was used for example in such commercial web-sites as Epinions and Amazon (according to the research of Jøsang et al. in 2007 [JIB07]) or in the system of Jurca et. al [JuF03].

Furthermore, reputation can be represented as a weighted average of all the experiences. In this case different factors (e.g., age of the experience and source credibility) can be taken as the weights. This method was used for example in FIRE system [HJS06] and knot-aware trust-based reputation model [Gal11].

Discrete models imply that reputation or trust can be represented in terms of discrete values. Discrete values can be for example: Very Trustworthy, Trustworthy, Untrustworthy and Very Untrustworthy [JIB07]. These values are further used for example for subjective defining local trust in a trustee or referrer [AbH00]. Discrete trust models include works of Cahill et. al [CGS03] and Yu et. al [CNS03].

Probabilistic models represent reputation score either as a probability of for example positive outcome or as a set of probabilities for different possible outcomes (e.g., positive, negative and unknown effect) which add up to 1. Probability can represent either a frequency of a certain outcome, degree of belief in a trustee's rationality or plausibility of a statement [Ruo12]. Multiagent systems especially use probability representation of reputation [Ruo12]. The main reason for this is that probability representation of reputation is not complex yet quite representative and flexible, taking into account the number of agents and differences between them in a multiagent system.

In TuBE [Ruo12], which focuses on trust decisions in inter-enterprise collaboration, risk is represented as a set of probabilities of different outcomes for every asset (monetary, reputation, control and satisfaction) guarded by a trustor enterprise. Managing the Dynamic Nature of Trust (MDNT) [SBL04] predicts a trustee's behavior proba-

bilistically based on the experience from a specific time period. Maximum Likelihood Estimation of Peer's Performance (MLE) [DeA04] uses a probabilistic approach to identify the probability that a referrer can give an incorrect information.

Probability distribution models view results of transactions as a sample of binomial or multinomial distribution and apply Bayes' theorem to define the required parameters of the selected distribution [Ruo12]. For this reason such models are also referred as Bayesian reputation models [Ruo12].

Reputation management systems of this kind usually apply the Beta probability distribution (e.g., [JøI02], [YuS02] and [TPJ06]). The Beta distribution reveals two challenges for reputation management algorithms [Ruo12]. First, as it is a distribution of binary events, the format of reputation representation is limited to binary (e.g., a trustee behaves with a trustor either good or bad). Second, as ordering of events on the distribution level does not affect the parameters of the distribution, dealing with dynamism of information (e.g., its aging) should be additionally considered in the reputation management algorithm.

Beta distribution is a subset of Dirichlet distributions, which are also used in reputation management systems (e.g., [JøH07] and [RRR07]). These distributions can represent reputation as multiple discrete outcomes [Ruo12].

Belief models are connected with probability theory and can represent the set of probabilities of the possible outcomes that do not necessarily add up to 1, assigning the remaining value to uncertainty [Gal11]. For example reputation of an agent can be represented as a triple $\langle b, d, u \rangle$, representing *belief* (probability of a positive outcome), *disbelief* (probability of a negative outcome) and *uncertainty* respectively; $b, d, u \in [0, 1]$ [JIB07]. Thus, the sum of probabilities of positive and negative outcomes does not necessarily equals to 1, whereas $b + d + u = 1$.

Yu et. al use belief models to represent reputation in belief space [YuS02]. However, it is more common to use belief models for trust representation (e.g., Jøsang [Jøs01], Paradesi et. al [PDS09] and Wang et. al [WHS11]).

A **fuzzy logic** membership function is used to identify to what extent an agent can be described as for example trustworthy or untrustworthy [JIB07]. The REGRET reputation system [SaS02], as well as the scheme proposed by Manchala [Man98] apply such models.

Flow models compute reputation or trust based on looped or arbitrary long chains of agents [JIB07]. In terms of graph theory, the network of agents can be represented

as a directed graph, where the weights of arcs represent reputation of an agent or trust placed in it. Generally, an agent's reputation increases as a function of incoming arcs and decreases as a function of outgoing ones. To represent trust in binary terms the model focuses on the existence of the arcs, leaving their weights out. In case there is an arc from agent A to agent B, agent A trusts B, otherwise agent A does not trust B.

In some systems (e.g., Advogato [Lev09] and Appleseed [ZiL04]) there is a constant trust or reputation weight assigned to the whole community. Agents distribute and redistribute trust or reputation weights between each other. In other systems (e.g., EigenTrust [KSG03]) there is no constant trust or reputation weight for the whole community, however the values are iteratively or repeatedly computed till they are converged to stable values.

If a reputation management system is not incorporated in a trust management system, it should specify an algorithm for making trust decisions. In other words it should specify an algorithm of choosing a trustee based on the gathered reputation information. A trustee can be either a service provider to collaborate with or a referrer to request information from. Making trust decisions and performing transactions is important for a reputation management system, for it to update reputation values based on the experiences between parties.

The most commonly used method to identify trust placed in an agent in a reputation management system based on a **threshold** [RKK07]. Trustor specifies a threshold of reputation value, and if a trustee's reputation meets the requirements, the trustor interacts with it.

Ranking is another method of taking a trust decision [RKK07]. A trustor ranks trustees according to their reputation values (e.g., local or overall) and based on this ranking chooses one to interact with [RKK07].

3.2.3 Vulnerabilities of reputation management systems

There are common problems identified for all the reputation management systems. In other words, these systems contain vulnerabilities that can be used by a trustor, referrer or a service provider for undermining or bypassing a protection offered by the system [KeC09]. A trustor can use vulnerabilities of a reputation system while either rating a service provider or later providing its ratings to other agents, performing a role of a referrer. For this reason we group system vulnerabilities from a trustor's

and referrer's points of view, representing a group of system vulnerabilities from the point of view of a referrer. Namely, if a trustor does not provide its ratings further, we assume that it does not take advantage of system vulnerabilities and thus does not perform any attacks.

Attack is a sequence of events with a desired outcome, that is performed at the level of system usage and aimed to exploit vulnerabilities of the working logic of the system [KeC09]. An attack can target at multiple vulnerabilities.

Below we present possible vulnerabilities of reputation management systems from a referrer's and a service provider's point of views. Moreover, we highlight most common problems of reputation management systems. We present also some solutions that are proposed by different authors to deal with the specified vulnerabilities and problems. Knowledge of common problems of reputation management systems will allow us to broaden the comparison of reputation management algorithms in our system.

Possible system vulnerabilities from the point of view of a referrer are: unfair referrals and discrimination, and ballot box stuffing. Below we have a closer look on these vulnerabilities.

As sincerity of referrals provided by referrers can not be controlled in any way, there arises a common problem of obtaining possibly **unfair** positive or negative **referrals**. Moreover, referrers can **discriminate** some agents over others, providing fair referrals to the former and unfair to the latter.

Jøsang et al. group the proposed solutions against this vulnerability into two groups: endogenous and exogenous discounting of unfair referrals [JIB07]. Endogenous discounting of unfair referrals implies rating of the gathered referrals or referrers themselves based on their referrals. Ratings of the referrals or referrers are identified by the statistical properties of the referrals.

Exogenous discounting of unfair referrals implies rating referrers based on their reputation as a service providers. The assumption here is that a service provider with good reputation is likely to provide fair referrals, while a service provider with bad reputation is likely to provide unfair referrals. Moreover, rating of a referrer can be done based on comparison of local information about the required service provider with the referrer's information.

Ballot box stuffing implies that a referrer submits more than required number of referrals that can be unfairly positive or negative for a service provider [JIB07].

Such vulnerability can appear in reputation management systems, where referrals are submitted to a store. A reputation management system can protect against this behavior by controlling a way referrals are provided after completion of transaction. In case referrals are provided upon request the ballot box stuffing problem does not appear, as a referrer can only provide one referral per request and the content of the referral later can affect the referrer's reputation. However, a referral still can be unfair, in which case we refer to the previous vulnerability.

Possible system vulnerabilities from the point of view of a service provider are: "re-entry" problem, behavior variations over time and "exit" problem, value imbalance, fake transactions, discrimination, low incentives for submitting referrals, and bias towards positive referrals. Below we have a closer look on these vulnerabilities.

"Re-entry problem" implies the following. Reputation management systems assume that agents of a multiagent system are long-lived, which makes possible for a reputation algorithm to identify a service provider's (or referrer's) reputation based on its past actions. However, it is easy to register a new identity in a multiagent system. For this reason it may be in a service provider's interests to register itself as a newcomer, if previously it gathered poor reputation. It is not in the interests of the entire community though, as agents are expecting a reputation management system to be effective. This vulnerability implies that a service provider can improve its reputation without even being involved in any transaction [KeC06].

Possible solution to such "re-entry" problem includes penalizing newcomers (e.g., financially). However, such approach can discourage "good" newcomers from joining the system [JIB07].

Behavior variations vulnerability implies the following. The quality of services can vary over time either due to the deliberate policy of a service provider or due to uncontrolled factors [JIB07]. These quality variations lead to the changes in a reputation of service providers in such a way that when the quality has changed, erroneous previous reputation will still be relied on for some time lag. It applies as well to the **"exit" problem** [KeC06], when a service provider wants to leave a network. From the point of time when decision to leave a network is taken till the actual leaving a service provider can still use its previous good reputation to cheat on other agents of the system. After deciding to leave the network, the service provider is not interested in keeping its good reputation. Instead, it can take advantage from this leaving.

Possible solutions to the behavior variations and "exit" problem include discounting

past referrals connected with a particular service provider or specifying for referrals time to live [JIB07]. Discounting past referrals can be based on time, frequency of transactions or combination of both. However, discounting past referrals involves storing each referral with a timestamp, which can be either costly or can steadily lose past information [Ruo12]. Moreover, as the main goal of identifying behavior changes over time is to quickly react to these changes (e.g., not allow good history of interactions to overweigh bad recent behaviour), time is not an effective measure to identify behavior changes [Ruo12]. Instead, reputation epochs can be applied [RHK11], where the measure of changes in behavior is the new information about service provider's behavior that was previously unknown. The number of reputation epochs shows the consistency of a service provider's behavior, while the weight assigned to the current epoch identifies how quickly the system reacts to the changes in a service provider's behavior.

Value imbalance vulnerability appears when reputation management systems do not bind a referral to the significance of the transaction that caused it [KeC06]. However there can be small and big transactions between entities. Small transaction can be for example financially cheap or not very important, while big transactions can be financially expensive or be a core of a business process. Thus, a service provider can gain good reputation performing some small transactions and then use this reputation to cheat other agents in big transactions. One possible solution to this value imbalance vulnerability is to bind referral with the significance of transaction that caused this referral.

Good reputation of a server provider attracts trustors. Thus, service providers are willing to obtain good reputation [KeC06]. As service providers can perform a role of trustor as well, they can organize **fake transactions** between each other to increase their reputation. This vulnerability points to the above-mentioned ballot box stuffing vulnerability from the point of view of a referrer, as service providers become referrers when providing their referrals in this case.

Service providers, similarly to referrers, can **discriminate** some agents over others, providing good service to the former and bad to the latter. While the proportion of discriminated agents is small, their opinion about service providers can be assumed as unfair by other agents. Thus, a reputation management system should have an algorithm for identifying discrimination behavior of the service providers or be able to identify their victims [JIB07].

One possible solution to this vulnerability is to make a collaboration contract every

time a trustor interacts with a service provider. After collaboration the trustor states whether the relevant contract was followed or not, supporting its statement with the non-repudiable receipts [RKK12].

Common problems of reputation management systems include low incentives for submitting referrals and bias towards positive referrals. Below we have a closer look on these problems.

Problem of **low incentives for submitting referrals** implies the following. After collaborating with a service provider a trustor should either submit its referral to a store, or provide it only when other agents request. In both cases incentives for providing a referral are low, as in a generic scenario there are no rewards for doing that or punishments otherwise. Along with a number of other possible reasons, that may result in not submitting a referral. Other reasons may include for example general positiveness of a referrer and unwillingness to spoil reputation of any service provider; or if a service provider has limited capacity, referrer may be not interested in attracting extra clients to this server provider by providing its good referrals.

In order to solve this problem, specific incentive mechanisms can be considered in the reputation management systems. Miller et al. and Jurca et al. propose incentive mechanisms based on financial rewarding or connected with other financial interests of the referrers [MRZ02, JuF03]. Ruohomaa et al. propose making it a part of the collaboration contracts [RKK12].

As referrers tend to provide mostly positive referrals, there appears **bias towards positive referrals** problem. It can be explained for example by unwillingness to spoil reputation of others or by expectation of obtaining good referrals in return.

In order to solve this bias problem, Ismail et al. suggest providing anonymous information [IBJ03]. However, in this case it will be difficult to rely on such information, and there will be no way to define its credibility and extent to which it can be trusted.

It is specific to a reputation management algorithm whether it considers possible vulnerabilities and common problems for such systems or not. Our model includes these solutions as long as they are built in a reputation management algorithm. Our model does not support any algorithm-independent solution to deal with possible vulnerabilities and common problems of such systems. Instead, we use this knowledge to broaden the comparison of reputation management algorithms in our system.

3.3 Testbeds for the comparison of reputation management algorithms

This section represents related work of the systems that aim to compare reputation management algorithms and exposes their drawbacks in comparison with our system. Below we present two testbeds for comparing reputation and trust management systems in a marketplace scenario: the ART and TREET [ART12, KeC09]. Moreover, we present a model for a testbed for evaluating and comparing reputation and trust management systems [ChE11].

3.3.1 The Agent Reputation and Trust (ART) testbed

The ART testbed simulates a market environment where different agents can pursue different trust strategies. The testbed aims at comparing these strategies [ART12]. In the ART service providers are presented as painting appraisers [FKM06]. Similarly to our system, they can play a role of either a trustor or a referrer between each other. Additionally to these roles, ART defines also a role of a client of a service provider. Clients can only request and obtain the service. Thus, only service providers share the reputation information.

Service providers evaluate paintings presented by clients as accurately as possible in order to increase the number of their clients. Paintings can belong to different eras, and service providers can have different expertise in these eras. Service providers are aware of their own expertise and are not aware of the expertise of other service providers. In case a service provider realizes its expertise is not enough for evaluating a painting, it can request a service from other service providers, becoming a client in this case. Service providers do not necessarily expose accurate service to other providers as their competitors. Thus, service providers can also request reputation information about each other.

The ART model adds monetary relationships to our model. The final aim of the simulation is to identify a service provider with the highest bank account balance and analyze its behavior (e.g., accuracy and consistency of the provided service).

While the testbed presents a reasonable tool for comparing trust strategies in a competitive environment, it does not suit our purpose of evaluating reputation management algorithms for several reasons. First, representation of reputation and quality of a service is limited, as the main focus is paid to the trust strategies. Our model, in contrast, places greater emphasis on flexible representation of reputation.

Second, when choosing a service provider a trustor can only request for a particular service provider's reputation from the other providers at a time, which limits the real world's settings. In other words a trustor cannot request reputation values of several service providers at a time, neither it can specify the service to collect the reputation values of all the service providers that other agents had interactions with concerning this service. Thus, a service provider can only go through the whole list of other providers in searching for an acceptable reputation of one of them in a particular era. In our model, in contrast, an agent specifies a particular service (e.g., painting era) about which it wants to gather information. Requested agents can provide reputation information of all the service providers they had experience with concerning the specified service.

Finally, the marketplace scenario and monetary relationships between the actors from our point of view add an overhead to the model, if its main purpose is to evaluate and compare different reputation management algorithms. In our model agents also provide services to each other, which can be argued to be a marketplace scenario as well. However, we assume that evaluation of reputation management algorithms can be based on the relationships of service providing without monetary and contract conditions, which can bias the evaluation results.

3.3.2 The Trust and Reputation Experimentation and Evaluation Testbed (TREET)

TREET is a testbed for evaluating reputation and trust management systems against various behavior attacks in the marketplace scenario [KeC09]. Service providers are presented as sellers, that can belong to honest, randomly cheating agents or those that pursue a special cheating strategy. Trustors and referrers are presented as buyers. So, the roles of a service provider and a buyer are separated, meaning for example that a service provider cannot perform a role of a buyer. Thus, only buyers share reputation information between each other.

In the scenario of the system service providers can advertise products from a fixed set. Trustors, based on reputation information, decide whether to buy products or not and from which service provider. After a service provider obtains payment from a trustor, it can either deliver a product or cheat on the trustor by not shipping the product within the system's equivalent of 14 days. The trustor after obtaining the product or realizing that he/she was cheated, updates his/her experience information concerning the service provider.

The testbed considers the following attack scenarios that can be performed by service providers [KeC09]: playbook, re-entry and proliferation attack, reputation lag attack, and value imbalance attack. Further we have a closer look on these attacks.

Playbook attack implies that a service provider has a "book" of "plays" or strategies, including sequence of malicious actions. At any given moment a service provider chooses a certain strategy and performs it.

In the re-entry attack a service provider creates its identity, uses it for cheating during some time and finally removes it to create the next one. In the proliferation attack a service provider creates different independent identities to offer its products through all of them.

The reputation lag attack implies the following. Let us suppose that the contracts of the multiagent system allocate 14 days for the products' shipping. In case of cheating during this time reputation of a service provider will not change, as the trustor can provide its referral only after obtaining the product. Thus, a service provider can behave honestly before shipping the product, then it starts to behave dishonest during the shipping time of 14 days, after which it removes its current identification and creates a new one.

The value imbalance attack implies that a service provider gains a good reputation behaving honestly in insignificant transactions and then cheats in performing more valuable transactions.

Different reputation and trust management algorithms were evaluated in the testbed (e.g., TRAVOS, Basic Trunits and the Beta Reputation System) [KeC09]. TREET testbed overcomes many of the ART limitations and presents a new more flexible platform for evaluation and comparison of trust and reputation management systems in a marketplace scenario. Features of TREET include for example adjustable and flexible representation of reputation and trust between agents, eased monetary relationships between agents, variety of considered attacks, ability to use different reputation and trust management systems between different agents in the same simulation run, and possible collusion behavior among agents [KeC10].

However, the TREET does not suit our purpose of evaluating different reputation management algorithms for several reasons. First, the environment still implements the pure marketplace scenario, including monetary relationships, which from our point of view narrows the application perspective of trust and reputation management systems. Thus, the effectiveness of a certain agent strategy is evaluated as the

sum of all the monetary gains and losses. Furthermore, agents are assumed to have unlimited budget, which is questionable for the real market environment.

Second, the testbed focuses on evaluating systems against different attacks from service providers. Thus, on the one hand, the testbed considers only attacks from service providers, assuming that buyers are perfectly honest. On the other hand, the testbed does not evaluate any other characteristics of the system, however it allows to plug in users' own tests that can aim at any characteristics of the system. Our model, in contrast, enables to specify behavior patterns for both service providers and referrers, making possible to set up dishonest behavior patterns for the referrers as well. Moreover, we evaluate a set of characteristics of reputation management systems that are meant to compare these systems in their intended, non-vulnerable environment (e.g., trustor's reactivity to changes in a behavior of a referrer and recovery of a service provider's reputation if it fails to provide a service for a while).

3.3.3 A model for a testbed for evaluating reputation systems

Chandrasekaran et al. introduce an application-independent model for a testbed for evaluating reputation systems and comparing them against different attacks [ChE11]. The model represents a generic workflow of graph transformations. The workflow includes feedback history graph, reputation graph and trust graph.

The feedback history graph includes all the feedbacks provided in the system. The graph represents a single feedback as a value in $[0, 1]$, indicating a referrer's satisfaction in a service.

Reputation algorithm of a reputation system defines the transformation from the feedback history graph to the reputation graph. The reputation graph in the model can represent either global or local reputation values. Reputation value is represented as a weight of an incoming arc. In the case of a global reputation graph, reputation values of an agent are the same. In other words, weights for all the incoming arcs are identical. In case of local reputation graph, the arcs' weights can be different from different agents. The model allows arcs also to close to the same agent, representing trust value that an agent places in itself for a certain task. The reputation graph in the model can be obtained either directly from the feedback history graph, or through an intermediate graph, which is built in some reputation algorithms [ChE11].

Trust algorithm of the reputation system defines the transformation from the repu-

tation graph to the trust graph. The trust graph represents boolean values of trust between agents. Thus, a directed arc from one agent to another appears only if the former agent trusts the latter.

Moreover, the testbed models slandering, behavior changes, and Sybil agents' attacks and evaluates systems' behavior against them. Slandering attack implies that an agent provides false negative feedback concerning another agent that may behave honestly. An attacker can also provoke other agents to give negative feedback towards the same victim. Behavior changes attack means that first an agent behaves honestly till it gains trust from other agents, after which it starts dishonest behavior. When dishonest behavior causes a decrease of trust from other agents, an attacker starts to behave honestly again. In the Sybil attack an agent creates fake identities, aiming at maximizing the number of the identities among them that are trusted by the other agents. Chandrasekaran et al. also include to their model metrics for evaluating a combination of Sybil with either behavior changes or slandering attack [ChE11].

Although the presented model is claimed to be generic, it reveals the following drawbacks. First, a feedback can be only represented as a single value. In all the systems presented as an example a feedback either can be identified as a single value (e.g., PeerTrust and ManagingTrust) or is not identified altogether [ChE11]. However, there are many reputation systems that represent feedback by more than one value. For example in Travos [TPJ06] a feedback includes two counters, representing positive and negative experiences. In Pinocchio [FKO04] every service provider is associated with the set of properties, that are later evaluated and communicated by the agents. Our model is similar to Pinocchio in this sense, providing a possibility of defining theoretically unlimited number of service dimensions that can be different for different services. Agents rate provided service of the service provider according to the service dimensions and supply the ratings as a referral.

Second, feedback information does not include timestamps, excluding the possibility of mapping reputation systems that contain temporal discount factors or other temporal information. For example the knot-aware trust based reputation model [GGH08] includes time intervals with their relative weights and changes over time. The maintenance-based trust model [KGB09] discounts past experiences with the timely relevance factor. Even though it is claimed that the model can be easily expanded with the timestamps [ChE11], the logic of the reputation graph construction will also need to be reconsidered. Our model includes timestamps (which may

be omitted, though) for reputation values of a service provider and a referrer. These timestamps may be used for discounting information or performing any other dynamism over the data.

Furthermore, the reputation graph can either represent global or local reputation values at a time, lacking possibility of representing both of them simultaneously. This means that in order to get full evaluation results of a system that considers both of these reputation values, we have to simulate a reputation system in the model twice with the same input characteristics. Chandrasekaran et al. presented examples using EigenTrust and PeerTrust systems that consider only global reputation values [ChE11]. Our model can contain local, external and subjective reputation values for a service provider or a referrer.

4 Tool for simulating reputation management algorithms

This chapter presents the main purpose and design solutions for our tool. To generic purpose of the tool is to simulate, evaluate and compare reputation management algorithm. However comprehensive evaluation of a reputation management algorithm includes various factors that are represented in Section 4.1. We identify our focus of the reputation management algorithm evaluation, which is the first and the main step for further development of the system. According to the identified purpose of the tool we further present system's architecture and design solutions.

4.1 Behavior evaluation of a reputation management algorithm

Behavior evaluation of a reputation or trust management algorithm consists of the following main parts [RuK13]:

1. Evaluation of the feasibility of the system, as well as other possible characteristics such as usability, portability or adjustability to different business situations.
2. Evaluation of trust or reputation update policy.
3. Evaluation of attack resistance of the system.
4. Evaluation of the system's incentives.

Evaluation of the system's feasibility includes for example scalability and efficiency analysis. Among other characteristics, efficiency analysis can include the storage load during the simulation process or speed of the algorithm's processing. Evaluation of trust or reputation update policy implies applying different representative loads to the system and tracking the results of trust or reputation evolution. Evaluation of attack resistance of the system implies applying different attack loads to the system and identifying the possibility for the attackers to succeed. Evaluation of the system's incentives implies evaluation of the system's ability to promote desirable behavior and eliminate misbehavior among agents.

Our current work focuses on evaluation of trust or reputation update policy. Evaluation of the system's feasibility can be performed by the users of the tool according to their particular interests. The possibility of the tool to support analysis of some characteristics (e.g. speed of algorithm's processing) is a future possible improvement to the system.

Evaluation of attack resistance of the system first of all includes constructing a comprehensive list of possible attacks from both service provider's and referrer's perspectives. Possible attacks are based on the vulnerabilities of the systems. Section 3.2.3 considers some of the possible systems' vulnerabilities. However, evaluation of attack resistance of the trust or reputation management system implies two main challenges. First, it is hard to identify all the possible systems' vulnerabilities and simulate all the possible attacks, as systems are quite diverse, and attackers can constantly come up with the new behavior patterns. Second, to represent the attack resistance of the system, possible actions of the attackers should be tied with cost of the impact of the action [RuK13]. The aim of the attackers then would be to maximize gains and to minimize losses. Then the attack resistance of the system can be evaluated by the extent to which attackers' aim is achieved.

Thus, evaluation of the attack resistance of the system implies introducing a level of monetary relationships to our model. Evaluation of the attack resistance of the reputation or trust management algorithm is out of scope of the current work and can become a future improvement of the system.

Evaluation of the system's incentives requires more research [RuK13]. However users of the tool can come up with their own evaluation method, as it does not require any special loads generation in the system.

Evaluation of trust or reputation update policy can be performed in two different scenarios: the agents follow the rules, and their behavior is expected; and the agent(s) follow the rules, however their behavior can be unexpected and possibly assumed as misbehavior. For both scenarios it is crucial to identify the rules of the system. For example there is a multiagent system that includes travel agencies as service providers. One of the accepted behavior patterns assumed in the system for the service providers can follow for example parabola pattern. So that when the customers' demand increases in summer time for example, the quality of travel agency's service goes down, while when the demand decreases in winter, the service quality goes up. This happens periodically according to the seasons.

The first scenario assumes that all the agents behave according to the specified behavior patterns. The second scenario assumes that agents of interest behave according to these patterns, but there are detected some rapid or unexpected changes in the pattern, which can be assumed as misbehavior. For example according to the parabola pattern in winter time the service of travel agency should be better than in summer. But travel agency can have a company's reorganization during winter which can lead to its service quality deterioration for a while. This behavior is still obedient, but can be assumed as misbehavior by other members of the multiagent system that do not know that this is a temporal sole action. However later the travel agency resumes its ability to follow the known behavior pattern in an expected way.

In the above described scenarios evaluation of trust or reputation update policy can be performed from the perspectives of a service provider or referrer. Figure 4.1 depicts the scenarios and perspectives of the possible evaluations.

		Point of view	
		Service provider	Referrer
Scenarios	Agents follow rules, behavior is expected	A	C
	Agents follow rules, behavior is unexpected	B	D

Figure 4.1: Evaluation of trust or reputation update policy.

Example evaluations that correspond to Section A in Figure 4.1 are: evaluation of the reputation evolution of the service provider that suddenly has changed its behavior; comparison of the reputation evolution of the service providers of the same service that pursue different service providing strategies; and evaluation of the reputation evolution of the service provider that provides a high-quality service but there is a burst of negative referrals concerning this provider.

Example evaluations that correspond to Section B in Figure 4.1 are: evaluation of the reputation evolution of the service provider that for a while fails to provide good service; and comparison of the reputation evolution of the service providers that provide the same service using different service providing strategies and fail to provide the expected service for a while.

Example evaluations that correspond to Section C in Figure 4.1 are: evaluation of the reputation of the referrer that pursues a certain strategy; and evaluation of the reputation of the referrer that suddenly changes its behavior.

Example evaluations that correspond to Section D in Figure 4.1 are: evaluation of the reputation evolution of the referrer that fails for a while to provide true referrals, providing false instead; and comparison of the reputation evolutions of the referrers that pursue different referring strategies concerning the same service for different service providers, but fail for a while to provide expected referrals according to the pattern.

Current work focuses on the evaluation of the reputation update policy from a service provider perspective, which corresponds to Sections A and B in Figure 4.1. Evaluation of the trust or reputation update policy from a referrer's perspective is a possible future improvement of the system.

4.2 Tool architecture and interface

This section presents environment and assumptions considered for the system's implementation. Moreover, it presents information flow of the system with the detailed look on the input configurations and possible configurable reports.

4.2.1 Assumptions for the system

Here we present architecture and interface of our tool for simulating reputation management algorithms. Architecture of the tool is based on our notation of trust and reputation management concepts presented in Chapters 2 and 3. Here we summarize our terminology, assumptions for a multiagent system, assumptions for an agent in a multiagent system, and general scenario of an agent's behavior in a multiagent system.

Agent in a multiagent system can perform one of the three different roles at a time: trustor TR, service provider SP or referrer RF. Trustors obtain services S from service providers and referrals R from referrers. There are two main evaluation parameters that can be calculated either referring to a service provider or to a referrer: reputation and trust. While calculating these parameters we can take into account credibility of the information obtained for these calculations. First we identify reputation value with the possibility of considering credibility of the information used for this calculation. An agent can identify reputation values separately for referrers and service providers. Further based on the reputation values and optionally overall credibility of these reputation values agent identifies its trust either in a referrer or in a service provider. Trust values are required for choosing a party to collaborate with

in a given context. For this reason trust values are not stored in the system. They are calculated when it is required. Based on the calculated values an agent chooses a party to collaborate with. After the collaboration reputation values are updated to consider the experience for further deals. Figure 4.2 schematically depicts a notion of multiagent system assumed in our tool.

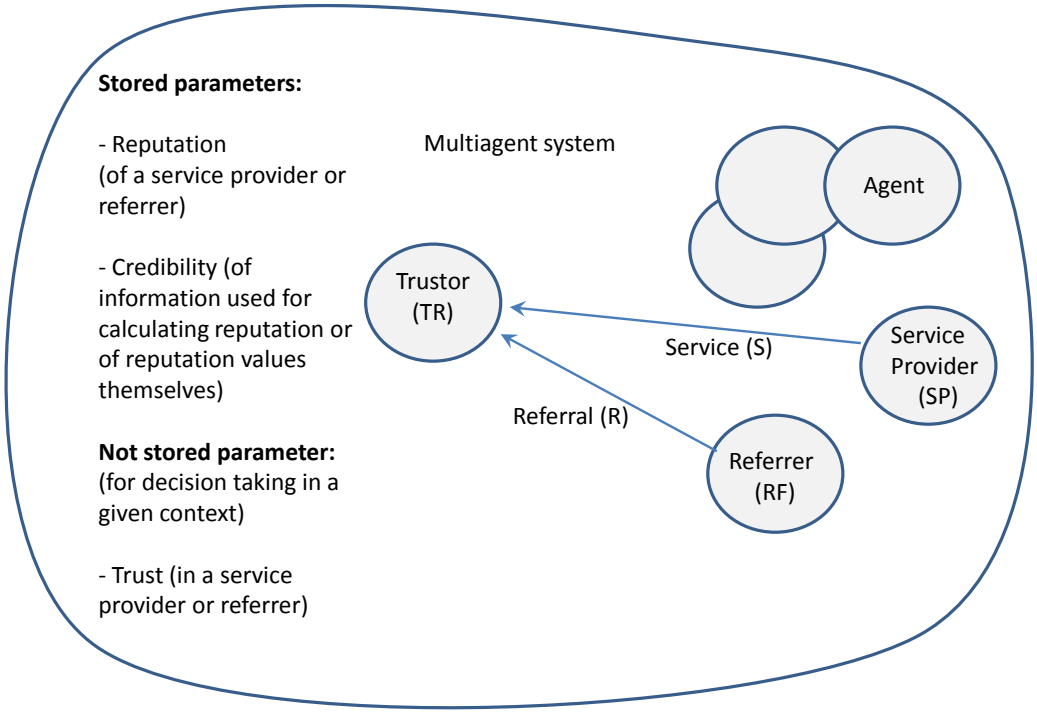


Figure 4.2: A notion of multiagent system used in our tool.

By default the tool stores local, external and subjective reputation values. Depending on the algorithm, external reputation value can be represented as a global reputation. A trustor TR_i keeps information about any other agent x from two different points of view: as a service provider and referrer. A trustor identifies the following parameters for any service provider x in a multiagent system: local reputation of x $LR_{SP}(TR_i, x)$; external reputation of x $ER_{SP}(TR_i, x)$; and subjective reputation of x $SR_{SP}(TR_i, x)$. A trustor identifies the following parameters for any referrer x in a multiagent system: local reputation of x $LR_{RF}(TR_i, x)$; external reputation of x $ER_{RF}(TR_i, x)$; and subjective reputation of x as a referrer: $SR_{RF}(TR_i, x)$.

For each of these reputation values the system can store credibility value, as well as credibility value can be stored for any parameter used in calculation of these values. That is, the system allows specifying any number of parameters used for calculating

reputation values, stores values of these parameters, stores credibility information for any of these parameters, and stores all the above-presented reputation values and their credibility information.

There are four assumptions for our model of a multiagent system:

1. There is no central trusted authority, no central server.
2. Agents are autonomous or behave as if they are autonomous. This assumption implies that even if agents are not autonomous (e.g., they are created by the same entity), the purpose of their multiple creation is for them to behave autonomously (e.g., a service provider creates another entity to hide its previous bad reputation or to try another behavior pattern). The tool does not support any algorithm-independent features for tracking down such behavior.
3. Agents are identified prior to simulation and cannot arbitrary leave or enter the system.
4. Agents are heterogeneous. They can be either real (humans or enterprises) or virtual.

There are three following assumptions for an agent in our multiagent system: agents use the same trust and reputation management system; agents' behavior is not different towards different participants; and agents' behavior type can change at any point.

Algorithm 4.1 presents general scenario of an interaction from a trustor's point of view in a multiagent system. From a referrer's point of view whenever it gets a request about service S, it checks its local database. In case it obtained S with any number of service providers greater than zero, it sends its local reputation values for all the found service providers. From a service provider's point of view whenever it gets request about particular service S, it provides it with the specified pattern for the service providing.

Algorithm 4.1 General scenario of an interaction from a trustor's point of view

Input: $LR_{RF}(TR, RF_i) = 0;\ i = 1, 2, ..., N;\ i \in \{1, 2, ..., m\}$
$LR_{SP}(TR, SP_j) = 0;\ j \in \{1, 2, ..., n\}$

Output: $LR'_{RF}(TR, RF_i);\ i = 1, 2, ..., N;\ i \in \{1, 2, ..., m\}$
$LR'_{SP}(TR, SP_j);\ j \in \{1, 2, ..., n\}$

1: TR desires service S.
2: TR requests all the agents about available information about service S.
3: TR receives referrals which contain reputation values from m referrers about n service providers that can provide service S, $RR_{SP}(RF_p, SP_q)$, where $p = 1, 2, ..., m$, $q = 1, 2, ..., n$.
4: TR identifies its trust in referrers, $T_{RF}(TR, RF_p)$ and deals only with those which it trusts above some threshold or according to the specific algorithm of the reputation management system.
5: TR adjusts referrals according to their credibility value, which is local to TR, $Lc_{RF}(TR, RF_p)$.
6: TR collects referrals information about every service provider SP_q into external reputation value $ER_{SP}(TR, SP_q)$.
7: TR combines credibility values into external credibility value $Ec_{SP}(TR, SP_q)$.
8: Based on $LR_{SP}(TR, SP_q)$ and $ER_{SP}(TR, SP_q)$ TR calculates subjective reputation value for every SP_q, $SR_{SP}(TR, SP_q)$, taking into account credibility values if algorithm supports it (in case credibility value c lies in $[0, 1]$, credibility value of local information Lc is always 1).
9: TR calculates its trust value placed in every service provider $T_{SP}(TR, SP_q)$.
10: Based on the trust values, TR chooses one service provider SP_j which it trusts the most in the given context.
11: After obtaining the service, TR updates its local and subjective reputation values of the service provider, $LR'_{SP}(TR, SP_j)$ and $SR'_{SP}(TR, SP_j)$
12: Based on the new subjective reputation value TR updates its local reputation values for the referrers that provided referrals for SP_j. That is, TR updates N local reputation values of referrers, $LR'_{RF}(TR, RF_i);\ i = 1, 2, ..., N;\ i \in \{1, 2, ..., m\}$.

Algorithm 4.1 is a generic algorithm of a trustor's behavior. Reputation management algorithms can omit different steps of the algorithm. However, reputation management algorithms differ mainly in the calculation methods of reputation, trust and credibility values.

4.2.2 Information flow in the system

Figure 4.3 presents information flow in the system. The simulation module is considered as a single block here. The information flow in the simulation module is not of an interest in the diagram. Instead, we aim to illustrate the broaden information flow that includes input and output modules for the whole system.

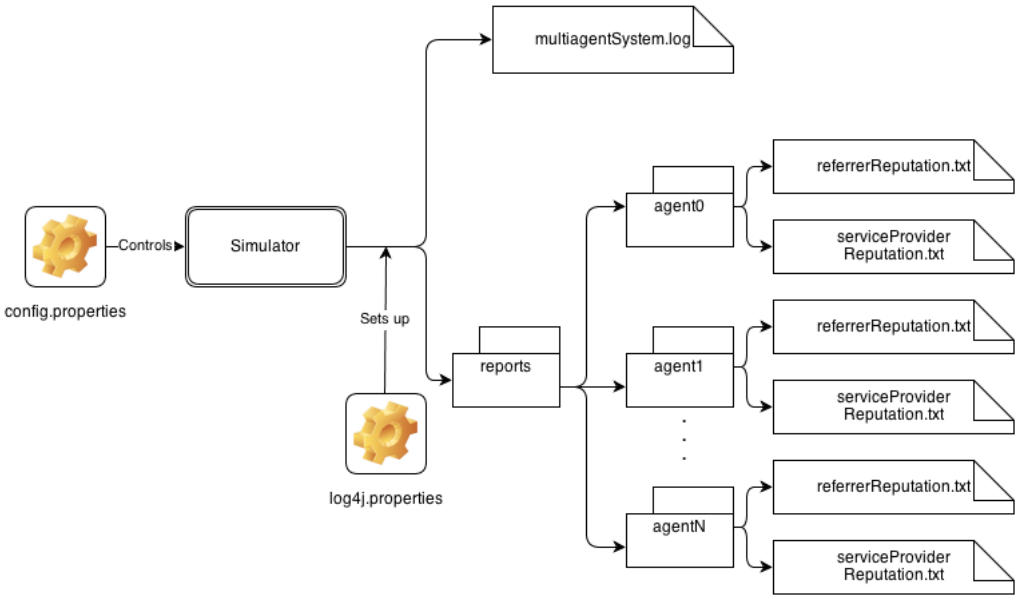


Figure 4.3: Information flow in the system.

Config.properties file serves the purpose of user 'interface' for configuring the simulation process. Simulation system then simulates a multiagent system behavior according to the specified parameters. As a result of simulation process user get a main log file multiagentSystem.log which logs every step of the simulation process. Moreover, there are generated text files that combine all the required information for the graphical reports. Graphical reports are configured in the config.properties file, representing interests of the user. Whenever system's servlet is started, the web page with the url of the servlet displays all the options for graphical reports that can be generated by user at any moment of the simulation process.

Configuration of the system in the config.properties file includes the set up of following aspects: general environment; system's simulation behavior; the plugged-in reputation management algorithm; and system's reports. Further we take a closer look on these configuration aspects.

General set up includes defining rate of the service requesting in the system; defining

list of all the distribution patterns that are used in the simulation; defining list of all the possible services that agents can provide in the system; and specifying the number of agents in the system.

Setting up system's simulation behavior includes assigning request patterns for all the services of the system; identifying service providers in the system and whether they can be trustors themselves or not; and assigning patterns for service providing. Below we take a closer look on the configurations for setting up the system's simulation behavior.

Request pattern for a service should be one of the patterns specified in the general set up. Request patterns for services can be assigned in an absolute manner, i.e. request patterns are mapped to the service ids. Otherwise request patterns can be assigned to the percentage of the services.

To identify service providers in the system we need to assign services to the agents that are assumed to be service providers. Services can be assigned in an absolute manner, i.e. services' ids are mapped to the agents' ids or service providers' ids. Otherwise services can be assigned to the percentage of service providers. In the latter case we first specify the percentage of all the agents that are service providers.

We can also assign patterns for service providing in an absolute or relative manner. To assign a service providing pattern for a service provider in an absolute manner, we need to map service id and pattern id to the service providers' ids. Otherwise a service providing pattern can be mapped to the percentage of service providers, i.e. service id and pattern id are mapped to the percentage of service providers.

Set up of the plugged-in reputation management algorithm tells the system which reputation management algorithm to use and what are the parameters values needed for this algorithm. Prior to that set up, the algorithm should have been implemented in the system or plugged into the system.

Setting up system's reports include specifying service providers whose reputation analysis is of interest and specifying groups of the service providers whose reputation values comparison is of interest.

According to the specified system's configurations, simulator models the overall system's behavior, 'telling' to the right agents their actions at a given rate. Moreover, at a run-time the simulator stores in the separate text files the information that is required for the specified reports.

5 Representation of the example algorithm in our system

This chapter describes the probabilistic approach for maintaining trust based on evidence [WHS11] according to our notation presented in Chapter 2. Moreover, it presents implementation of the algorithm in our system. The probabilistic approach for maintaining trust based on evidence was chosen for several reasons. First of all, the algorithm presents cutting edge research of the trust and reputation management algorithms. Thus the algorithm is quite comprehensive and considers many features that are only partly implemented in other algorithms [WHS11]. For this reason a lot of other algorithms can be described and implemented in the similar way. Second, from our point of view the algorithm presents one of the biggest and most interesting group of reputation and trust management algorithms using probability for the reputation update.

We have implemented the probabilistic approach in the system as a separate package. Appendix A represents some implementation decisions for the probabilistic algorithm in our system. There are four main classes that are used in the algorithm. Table A.1 describes these classes. Table A.2 presents main methods of the API for the algorithm.

5.1 Main concepts of the algorithm

The probabilistic approach for maintaining trust based on evidence [WHS11] considers agents in a multiagent system, specifying the following possible roles: service provider SP, referrer RF, and a trustor TR. In the approach a trustor maintains its trust historically, in other words in service providers, and socially, in other words in referrers. The approach represents trust placed in a service provider, which according to our notation is actually representation of the service provider's reputation. Similarly, representation of trust in a referrer in the approach maps to our representation of the referrer's reputation. However, the approach does not specify an algorithm for making trust decisions. In other words, it does not specify an algorithm for choosing a trustee based on the gathered reputation information. In Section 3.2.2 we specified common algorithms for making trust decisions - either using a threshold or ranking the trustees. Here we apply the ranking method. We rank trustees according to their overall reputation values and choose one of them

with the highest rank. In case there are several possible alternatives, we choose one of them randomly. The probabilistic approach does not support calculation of credibility of reputation values.

In comparison with our model for a multiagent system the probabilistic approach broadens possible actions of the referrers by introducing the following scenario. If a referrer does not have any direct experience with a service provider, it can point to another referrer that potentially has such experience. Thus a trustor for example asks about a service S_1 from three of its neighbors RF_1, RF_2, and RF_3. It obtains three replies: RF_1 and RF_2 refer to one service provider SP_1, while RF_3 refers to another service provider SP_2. After aggregating information about service providers from different sources and its own experience, the trustor decides for example to collaborate with SP_2. After interaction with SP_2 the trustor updates local reputation of SP_2, as well as reputation of RF_3. However, to enable the aforementioned scenario the trustor should also store the fact that referrers RF_1 and RF_2 have direct experience with SP_1. Our model does not consider this scenario, assuming that a referrer can only provide a referral of a service provider with which it had direct experience. However, the scenario can be introduced in the model in future work.

5.1.1 Reputation representation

Wang et al. define reputation in terms of two dimensions: probability and certainty of a good outcome [WHS11]. Certainty of a good outcome is introduced to differentiate equal probabilities obtained out of completely different experiences. Thus, probabilities of a good outcome are equal if we have experienced one good outcome out of two interactions and 100 good outcomes out of 200 interactions. However intuitively reputation of the service provider based on these experiences should not be identical. Thus, certainty is important for measuring the amount of information which is used for defining reputation. Wang et al. base the certainty computation on the following assumptions: with the fixed probability and increasing total number of experiences, the certainty should also increase; and with the fixed total number of experiences and increasing number of negative experiences within them, certainty should decrease [WHS11].

The probabilistic approach represents reputation in both the evidence and belief spaces [WaS06]. In the **evidence space** reputation is represented as a binary event $\langle r, s \rangle$. Here, $r \in R, r \geq 0$ is the number of positive experiences; $s \in R, s \geq 0$ is the number of negative experiences of a trustor with a trustee. The total number of

experiences $t = r + s \geq 0$. Wang et. al. claim that although the number of positive and negative experiences can be represented as natural numbers including zero, it is important to bear in mind the possible need for discounting the reputation values (e.g., due to their aging) or introducing other required dynamism for them [WaS10]. For this reason, number of experiences are represented as real numbers.

The anticipated value of a probability α of a positive outcome is represented in the following way:

$$\alpha = \frac{r}{r+s} = \frac{r}{t} \tag{1}.$$

In case $r + s = 0$, it can be considered as 0.5 [WHS11].

The certainty based on the binary event $\langle r,\ s\rangle$ is represented in the form:

$$c(r,s) = \frac{1}{2}\int_0^1 \mid \frac{x^r(1-x)^s}{\int_0^1 x^r(1-x)^s dx} - 1 \mid dx \tag{2}.$$

Thus, $c(r,s) = 1$ in the case of perfect knowledge and $c(r,s) = 0$ in the case of complete ignorance. Based on (1), r can be represented as $t\alpha$ and s - as $t(1-\alpha)$. Thus we can represent $c(r,s)$ also in the following way:

$$c(r,s) = c(t\alpha, t(1-\alpha)) \tag{3}.$$

Thus, when α is fixed, certainty becomes a function of t, $c(t)$, while when t is fixed, certainty becomes a function of α, $c(\alpha)$.
In the **belief space**, a binary event is modeled in the form $\langle b,\ d,\ u\rangle$, where $b,\ d$ and u represent probability of a positive outcome, probability of a negative outcome and uncertainty, correspondingly. Each of $b,\ d$ and u is a real number in $(0,1)$ and $b + d + u = 1$ [WaS07].

Reputation values can be transferred between the evidence and belief spaces. We transfer reputation value from the evidence to belief space in the following way:

$$b = \alpha c = c\frac{r}{r+s},\quad d = (1-\alpha)c = c\frac{s}{r+s},\quad u = 1 - c \tag{4},$$

where c is the certainty based on the binary event $\langle r,s\rangle$. Further, given $b = \alpha c$ and $d = (1-\alpha)c$, we represent α in the belief space as follows:

$$\alpha = \frac{b}{b+d} \tag{5}.$$

To transfer reputation value from the belief to evidence space, we have a fixed α value, given in (5). Thus, certainty in (3) becomes a function of t and we need to find such t that $c(t) = 1 - u$. Algorithm 5.1 finds t, and further $\langle r,s\rangle$, given

$\langle b, d, u \rangle$ [WaS07]. We assume $e > 0$ is a necessary precision and $t_{max} > 0$ is the maximum number of possible experiences.

Algorithm 5.1 Calculation of $\langle r, s \rangle$ given $\langle b, d, u \rangle$ [WaS07]

1: $\alpha = \frac{b}{b+d}$
2: $t_1 = 0$
3: $t_2 = t_{max}$
4: $c = u - 1$
5: **while** $t_2 - t_1 \geq e$ **do**
6: $\quad t = \frac{t_1 + t_2}{2}$
7: $\quad$ **if** $c(t) < c$ **then**
8: $\quad\quad t_1 = t$
9: $\quad$ **else**
10: $\quad\quad t_2 = t$
11: $\quad$ **end if**
12: **end while**
13: **return** $r = t\alpha, s = t - r$

In our implementation of the algorithm, the default value for e is 0.01, for t_{max} is 1000. Below we explain our decisions.

As r and s represent the number of positive and negative experiences respectively, and in most cases these experiences can be represented as natural numbers, we do not need a high precision for these values. However, in case there is a need for performing any kind of dynamism over these data, the number of experiences can be represented as real numbers and precision of 0.01 should be enough for such purposes.

Value of t_{max} greatly affects the calculation results. For example we applied the algorithm to the reputation value in the belief space $\langle b, u \rangle = \langle 0.405, 0.448 \rangle$ with the precision set to 0.01 with different values of t_{max}. Thus if t_{max} equals to 100, 500 and 1000, the algorithm produces the following values for r and s respectively: $\langle 10.97, 3.98 \rangle, \langle 366.84, 133.15 \rangle$ and $\langle 733.69, 266.30 \rangle$. According to the results, with the growing value of t_{max}, values of r and s also grow. Our default value for t_{max} is set to 1000, which we suppose is enough for the number of maximum possible experiences between trustor and trustee. After reaching the limit of 1000, the counter is set to zero again. In this case reputation value in the belief space is saved and further calculations are based on this value.

5.1.2 Concatenation and aggregation of reputation

In order to obtain a subjective reputation value that a trustor TR_i places in a service provider SP_j given direct experience of TR_i with a referrer RF_n and a referral provided by RF_n for SP_j, Wang et al. suggest using a **concatenation operator** ($\otimes$) [WaS06]. The idea behind the operator lies in concatenating local reputation value that TR_i places in RF_n and local reputation value that RF_n places in SP_j. Given local reputation value of referrer that TR_i places in RF_n, $LR_{RF}(TR_i, RF_n) = \langle b, d, u \rangle$ and local reputation value of service provider that RF_n places in SP_j, $LR_{SP}(RF_n, SP_j) = \langle b', d', u' \rangle$, the subjective reputation value of service provider that TR_i places in SP_j equals

$$SR_{SP}(TR_i, SP_j) = LR_{RF}(TR_i, RF_n) \otimes LR_{SP}(RF_n, SP_j) = \langle b \cdot b', b \cdot d', 1 - b \cdot b' - b \cdot d' \rangle \quad (6).$$

Suppose there are N reputation values $LR_{SP}(RF_N, SP_j)$ provided by RF_N for SP_j, where N is a natural number. Let local reputation values that TR_i places in each of the referrers be $LR_{RF}(TR_i, RF_N)$. In this case in order to obtain a subjective reputation value that TR_i places in a service provider SP_j, Wang et al. suggest using an **aggregation operator** ($\oplus$) in the evidence space [WaS06]. The idea behind the aggregation operator lies in simply summing up available positive and negative experiences accordingly. Thus, given N concatenated reputation values between TR_i and referrers RF_N, $R_1 = LR_{RF}(TR_i, RF_1) \otimes LR_{SP}(RF_1, SP_j) = \langle r_1,\ s_1 \rangle$, $R_2 = LR_{RF}(TR_i, RF_2) \otimes LR_{SP}(RF_2, SP_j) = \langle r_2,\ s_2 \rangle$, ..., $R_N = LR_{RF}(TR_i, RF_N) \otimes LR_{SP}(RF_N, SP_j) = \langle r_N, s_N \rangle$, aggregated reputation value that TR_i places in SP_j equals $R_1 \oplus R_2 \oplus ... \oplus R_N$. Aggregation is performed in the following order: first R_1 and R_2 are aggregated resulting for example in R; then R_3 is aggregated with R, providing a new value for R; then in the same fashion all the rest values are aggregated with R, providing a new value for it. Aggregation of a pair of reputation values for example R_1 and R_2 is defined as follows:

$$R_1 \oplus R_2 = \langle r_1 + r_2, s_1 + s_2 \rangle \quad (7).$$

Below we illustrate the reputation representation, reputation concatenation and aggregation in the probabilistic approach [WaS10]. In this example we consider $t_{max} = 100$ and $e = 0.01$. Suppose there are two referrers Anne and Bill, one service provider Carl and trustor Dennis. Anne has eleven good and four bad transactions with Carl. That is, Anne's local reputation value of Carl in the evidence space is $\langle 11, 4 \rangle$. Bill has two good and eight bad transactions with Carl. That is, Bill's local

reputation value of Carl in the evidence space is $\langle 2, 8\rangle$. Suppose Dennis in the belief space has a reputation value of Anne as $\langle b, u\rangle = \langle 0.2, 0.5\rangle$, of Bill - $\langle b, u\rangle = \langle 0.9, 0.05\rangle$.

Given aforementioned conditions we first obtain reputation values in the belief space that Anne and Bill place in Carl independently. In Anne's case we have $t = 15$, $\alpha = 0.733$ according to (1), and $c = 0.552$ according to (2). Based on (4), we obtain the following reputation value: $\langle 0.405, 0.448\rangle$. In Bill's case we have $t = 10$, $\alpha = 0.2$, and $c = 0.522$. Further, we obtain the following reputation value: $\langle 0.104, 0.478\rangle$. Dennis performs the following steps in order to identify the reputation value of Carl:

1. Based on (6) Dennis concatenates reputation values of Anne and reputation value that Anne places in Carl. Thus, we obtain one concatenated reputation value that Dennis places in Carl: $\langle 0.081, 0.89\rangle$. Similarly, we obtain another concatenated reputation value based on Dennis' and Bill's experiences: $\langle 0.094, 0.531\rangle$.

2. Based on Algorithm 5.1, Dennis transfers above concatenated reputation values into the evidence space. That is, for Anne's report we obtain $\langle 0.501, 0.179\rangle$, for Bill's report - $\langle 1.443, 5.756\rangle$.

3. Based on (7), Dennis combines these reputation values, calculating the reputation value of Carl in the evidence space: $\langle 1.944, 5.935\rangle$.

4. Based on (2) and (4), Dennis transfers this reputation value into the belief space: $\langle 0.113, 0.541\rangle$.

With this example, we have illustrated both the concatenation and aggregation operations. Concatenation operation combines reputation value that trustor places in referrer and reputation value that the referrer places in the service provider. Aggregation operation combines concatenated results with different referrers into a single reputation value that the trustor places in the service provider. This value can now be used to make a trust decision for example by comparing it to some defined threshold. As the next step, we will look at updating the referrer's reputation based on how well their referrals have matched with the trustor's own observations in the past concerning particular service provider.

5.1.3 Update of referrer's reputation

Suppose there is a trustor, referrer and a service provider. The trustor identifies $\langle r_{RF}, s_{RF}\rangle$ reputation of the referrer, the referrer identifies $\langle r', s'\rangle$ reputation of the

service provider. After obtaining the service, the trustor defines an actual reputation value of the service provider as $\langle r_{SP}, s_{SP} \rangle$. Further, the trustor needs to update its reputation value of the referrer based on the comparison between referrer's communicated and actual reputation values of the service provider. The Algorithm 5.2 (lines 7-10) represents the general reputation update logic, which is common among various approaches [WHS11].

Algorithm 5.2 Update of referrer's reputation [WaS11]

Input: $\langle r_{SP}, s_{SP} \rangle, \langle r', s' \rangle, \langle r_{RF}, s_{RF} \rangle, \beta$

Output: $\langle r'_{RF}, s'_{RF} \rangle$

//Average-β

1: $\alpha = \frac{r_{SP}}{r_{SP}+s_{SP}}$

2: $c = \mathbf{c}(r_{SP}, s_{SP})$

3: $q = 1 - \sqrt{(\alpha - \frac{r'+1}{r'+s'+2})^2 + \frac{(r'+1)(s'+1)}{(r'+s'+2)^2(r'+s'+3)}}$

4: $p = 1 - q$

5: $c' = \mathbf{c}(r', s')$

6: $c' = cc'$

//General update

7: $\delta r_R = c'q$

8: $\delta r_S = c'p$

9: $r'_{RF} = \delta r_R + (1 - \beta) r_{RF}$

10: $s'_{RF} = \delta s_R + (1 - \beta) s_{RF}$

Instead of interpreting a referral from the referrer as either good or bad, it is represented as q good and p bad "referrals". Q in this interpretation is $0 \leq q \leq 1$, ranging from a completely inaccurate to a completely accurate referral; p is assumed to be $1 - q$. Thus, q defines the proximity between the referral $\langle r', s' \rangle$ and the actual reputation $\langle r_{SP}, s_{SP} \rangle$.

However, along with the accuracy of the estimation, we should also include the weight of the accuracy that depends on the certainty of the referrer. Thus, suppose we have two referrals $\langle 0, 1 \rangle$ and $\langle 0, 100 \rangle$ for a service provider, whose actual reputation is $\langle 10, 0 \rangle$ [WHS11]. Both referrals report that a service provider is not trustworthy, however the certainty of the first referrer is $c(0, 1) = 0.25$, while the certainty of the second one is $c(0, 100) = 0.99$. That is why the second referrer should be either punished or rewarded more than the first one, depending on the actual reputation of the service provider calculated by the trustor. That is, the weight assigned to the accuracy of the estimation should increase when the certainty of the

referrer increases. Further, we estimate each referral as $c'q$ good and $c'(1-q) = c'p$ bad "referrals". This is covered by lines 7 and 8 in the Algorithm 5.2. Moreover, previous experience $\langle r_{RF}, s_{RF} \rangle$ is discounted by its age, assuming β as a temporal discount factor (lines 9 and 10). Various approaches differ in terms of q calculation. In the probabilistic approach *Average-β* algorithm is used for updating the reputation value of a referrer. Lines 1-6 in the Algorithm 5.2 represent the *Average-β* approach.

Initial reputation values assumed in the probabilistic approach follow. Initial reputation value that the trustor places in the service provider $\langle r_{SP}, s_{SP} \rangle$ is set to $\langle 0, 0 \rangle$, indicating that the trustor has no prior experience with the service provider. Initial reputation value that the trustor places in the referrer $\langle r_{RF}, s_{RF} \rangle$ is set to $\langle 1, 1 \rangle$, indicating the trustor's willingness to consider the referrer's feedback. The trustor updates its reputation value placed in the referrer only after obtaining service from the service provider, thus leading to $r_{SP} + s_{SP} > 0$. In other words, in case $\langle r_{SP}, s_{SP} \rangle = \langle 0, 0 \rangle$ the trustor can not update its reputation value of the referrer.

We will now illustrate Algorithm 5.2. Suppose in terms of the previous example, after obtaining service from Carl, Dennis estimated his actual reputation value as $\langle 1, 0 \rangle$. In the evidence space, Dennis' reputation value of Anne equals to $\langle 5.34, 8.01 \rangle$, reputation value of Bill equals to $\langle 1894.73, 105.26 \rangle$. Suppose for example $\beta = 0.01$, then according to the Algorithm 5.2 updated reputation that Dennis places in Anne equals to $\langle 5.29, 7.93 \rangle$, in Bill - $\langle 1875.78, 104.21 \rangle$. Results show us that as Anne's report was somewhat accurate for both r_{SP} and s_{SP}, Dennis enlarged the reputation value of Anne by increasing r_{SP} and decreasing s_{SP}. Similarly, as Bill's report was somewhat inaccurate for r_{SP}, but to some extent accurate for s_{SP}, Dennis decreased both r_{SP} and s_{SP} in his reputation value of Bill.

In our tool value of β can be tuned by the users in the config.properties file. The default value is set to 0.01 according to usage by the authors of the algorithm [WHS11].

5.1.4 Update of service provider's reputation

Suppose a trustor has experienced $\langle r, s \rangle$ outcomes with a service provider, placing a reputation value of $\langle b, u \rangle$ in this experience. Suppose also that after interacting with the service provider, the trustor estimates the provider's current behavior as $\langle r', s' \rangle$. Given these conditions, the trustor needs to update its reputation value of the service provider, obtaining an updated reputation value $\langle b', u' \rangle$ for $\langle b, u \rangle$.

Algorithm 5.3 presents the *Average-α* algorithm used in the probabilistic approach for updating reputation of a service provider. Initial values for the algorithm are the following: $\langle b, u\rangle$ is set to $\langle 0.9, 0.1\rangle$, indicating that the trustor trusts its past experience with high confidence; $\langle r, s\rangle$ is set to $\langle 0, 0\rangle$. Reputation value $\langle b, u\rangle$ is updated only after the trustor's interacting with the service provider, resulting in $r' + s' > 0$.

Algorithm 5.3 Average-α: update of reputation of a service provider [WaS11]

Input: $\langle r, s\rangle, \langle r', s'\rangle, \langle b, u\rangle$

Output: $\langle b', u'\rangle$

1: $\alpha = \frac{r'}{r'+s'}$
2: $c = \mathbf{c}(r, s)$
3: $c' = \mathbf{c}(r', s')$
4: $q = 1 - \sqrt{\frac{\int_0^1 x^r(1-x)^s(x-\alpha)^2 dx}{\int_0^1 x^r(1-x)^s dx}}$
5: $b'' = b + cc'(1-q)$
6: $u'' = u + cc'q$
7: $\beta = \frac{b''}{b''+u''}$
8: $b' = r' + \beta r$
9: $u' = s' + \beta s$

Average-α compares past behavior of a service provider $\langle r, s\rangle$ with its current behavior $\langle r', s'\rangle$ [WHS11]. The reputation of the service provider increases if its current behavior is close to its past behavior. Otherwise, reputation of a service provider decreases. Thus the reputation of a service provider is based on the consistency of its behavior. For example if $\langle b, u\rangle = \langle 0.9, 0.1\rangle$, $\langle r, s\rangle = \langle 3, 4\rangle$ and we experience $\langle r', s'\rangle = \langle 4, 4\rangle$, then the updated value for $\langle b, u\rangle$ equals to $\langle 1, 0\rangle$. However, if we experience $\langle r', s'\rangle = \langle 3, 5\rangle$, $\langle b', u'\rangle = \langle 0, 1\rangle$. Temporal discount factor β is automatically computed in the algorithm, eliminating the need for its hard-coding or manual adjusting.

6 Evaluation and analysis

This chapter presents some evaluation and analysis examples that can be performed using our system. The reputation management algorithm that is used in the examples is the probabilistic approach for maintaining trust based on evidence described in Chapter 5.

The goal of our tool was to evaluate reputation update policy of the single algorithm or to compare reputation update policies of different algorithms. In the current work we have implemented only one algorithm that is why here we illustrate possible evaluations of the reputation update policy.

Reputation update policy can be evaluated from the perspective of either a service provider or a referrer. This chapter illustrates examples of the evaluation of the reputation update policy from a service provider's point of view. The logic of the evaluations from different perspectives stays the same. The differences are in the results of reputation update for a service provider or a referrer. However the purpose of the current work is not to evaluate comprehensively the chosen algorithm, instead we are aiming at illustrating our tool's features.

Examples of the possible scenarios for evaluation reputation update policy from a service provider's point of view are presented in Section 4.1. We illustrate the following scenarios:

1. Agents' reactivity to changes in the behavior of a service provider.
2. Recovery of a service provider's reputation if it fails to provide a service for a while.
3. Reputation evolution of different service providers of the same service that pursue different strategies.

The generic configurations to our evaluation runs for a multiagent system that uses probabilistic approach for maintaining trust based on evidence follow.

1. Number of agents in the system: 1000.
2. Service providers can be trustors themselves.
3. Rate of the services' requesting is 10 seconds.

4. There is one service that can be provided in the system. The service's domain is *IT*, the service's type is *Printing*. The service has one dimension that can be evaluated: *quality of paper*. The values of the dimension are: *bad, okay, good, perfect*.

5. Agent 0 is a service provider.

The configurations for the probabilistic approach for maintaining trust based on evidence are the same for all the evaluation runs. The configurations include:

1. Possible maximal number of experiences between two agents t_max=1000.

2. Precision of the calculations e=0.1.

3. Deviation to number of positive or negative experiences when updating service provider's reputation is set to *positive*.

4. Discounting factor when updating the referrer's reputation $beta$=0.01.

5. Threshold for trust decision is $belief$=0.8, $uncertainty$=0.2.

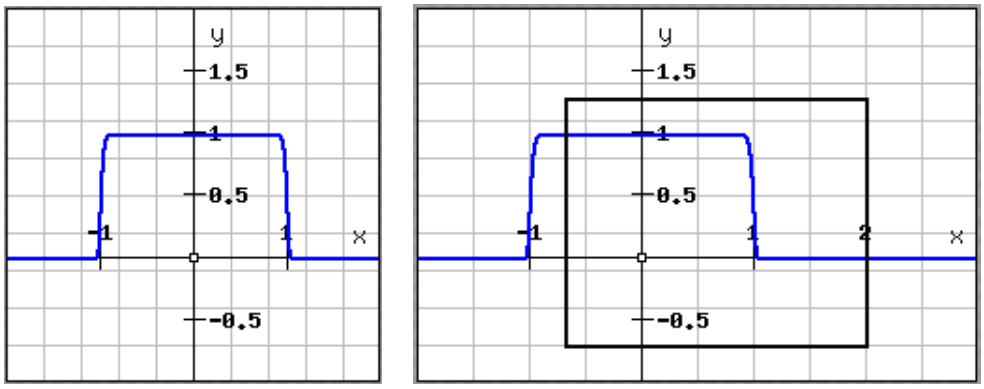
With the presented system's configurations, we will illustrate some system's features of analyzing reputation and trust management algorithms. Namely, we are going to analyze reputation update policy of service providers. We have chosen several scenarios for such analysis. Section 6.1 presents analysis of the agents' reactivity to changes in the behaviour of a service provider. Section 6.2 presents analysis of the recovery of a service provider's reputation if it fails to provide a service for a while. Finally, Section 6.3 presents analysis of differences in reputation evolution for different service providers that provide the same service but pursue different strategies.

6.1 Agents' reactivity to behavior changes

To evaluate agents' reactivity to changes in the behavior of a service provider, let us consider the pattern of the service providing as a Weibull curve, specified as a following function: $f(x) = e^{(-x^{50})}$. Figure 6.1(a) represents the distribution function.

We have chosen a power of x as 50 to make the pattern change sharper, so that the service provider's behavior changes quickly from the best to the worst. Let us choose a piece of the distribution that will be in use as a pattern for service providing from

$x = -0.6$ to $x = 2$. Figure 6.2(a) highlights the selected part. In this case we can track for some time the service provider's reputation evolution when it provides the best service only, as well as we can track the change in the reputation when the service quality rapidly worsens, and finally we can track reputation evolution when the service quality is the worst. Next let us consider the step for the service providing pattern. The bigger the step is, the quicker the round of the whole selected piece will take place. However at the same time the less amount of reputation updates will be tracked. To compromise on that, we have chosen a step of 0.07.



(a) Weibull curve. (b) Part of the Weibull curve in use.

Figure 6.1: Service providing pattern.

Let us choose a simple pattern for a service requesting, for example $f(x) = 3$. Thus, every 10 seconds 3 random agents will request the service one by one. After every interaction with the service provider its reputation will be updated and will affect the choice of the next agent. However for the representative results we have only one service provider so that every agent deals with this provider.

Appendix B represents parts of the multiagentSystem.log that reflect agents' reactivity to changes in the behavior of the service provider. Listing B.1 represents the multiagent system initialization Listing B.2 represents original best service providing pattern. Listing B.3 represents agents' reactivity to the rapid worsening of the service providing pattern.

Figure 6.2 represents the agents' reactivity in terms of belief and uncertainty values to changes in the behavior of the service provider. Figure 6.3 represents the number of positive and negative experiences with the service provider.

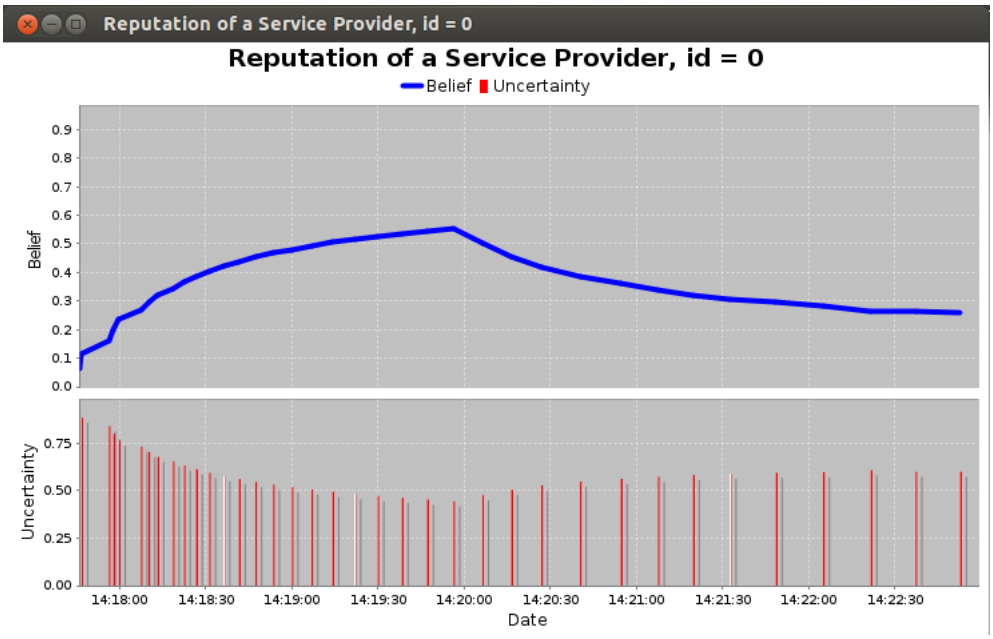


Figure 6.2: Belief and uncertainty evolution when service providing pattern is rapidly changed from providing the best service to providing the worst.

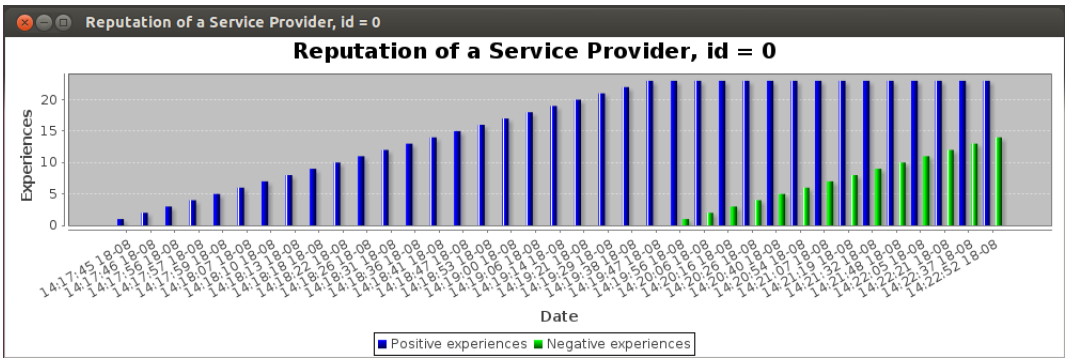


Figure 6.3: Number of positive and negative experiences with the service provider.

From $x = -0.6$ to approximately $x = 0.9$ service providing pattern is stable and service providing characteristics are the best. Number of positive experiences at this interval increases by 3 with every interaction with the service provider. However, we can see that the belief in the service provider is growing slowly, increasing by 0.55 after around 24 interactions. Uncertainty for this period decreases by 0.45 from 1 to 0.55. Then the service provider rapidly changes the behavior pattern. Belief in the service provider slowly decreases by 0.3 after next 14 interactions. Uncertainty for that period increases by around 0.15.

6.2 Recovery of a service provider's reputation

To illustrate evaluation of the recovery of a service provider's reputation if it fails to provide a service for a while we can use the same service providing pattern presented in Figure 6.1(b). First the service is provided with the best dimension value, then the service providing pattern is rapidly changed to providing the worst service. The worst service is provided from $x = 1$ to $x = 2$. With the step 0.07 and 3 service requests at a time, around 14 interactions will result in the negative experience with the worst service provided. After that the service will be provided again the best from $x = -0.6$ to $x = 0.9$. Figure 6.4 represents belief and uncertainty evolution when the service provider fails to provide the best service for a while. Figure 6.5 represents the number of positive and negative experiences with the service provider.

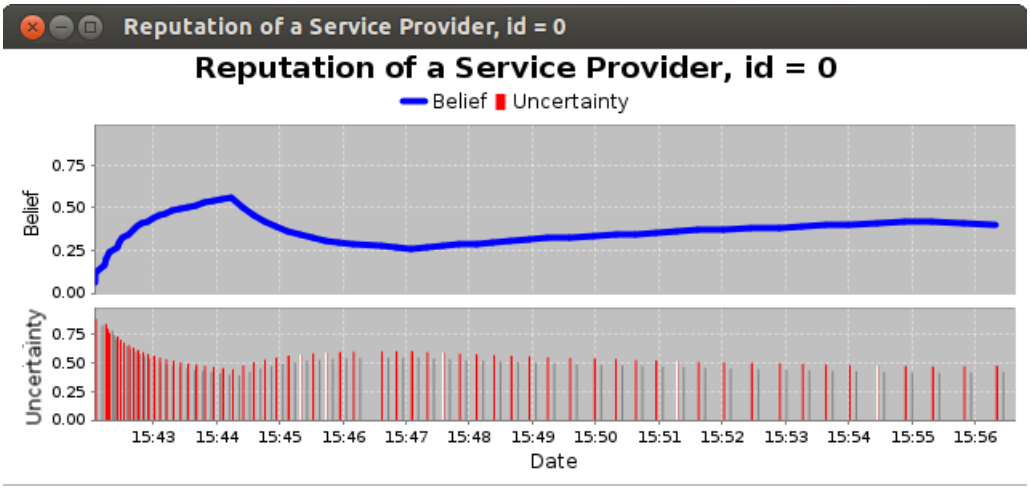


Figure 6.4: Belief and uncertainty evolution when the service provider fails to provide the best service for a while.

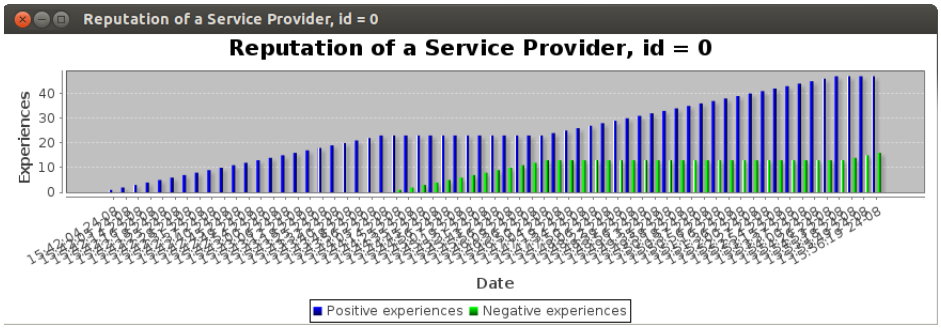
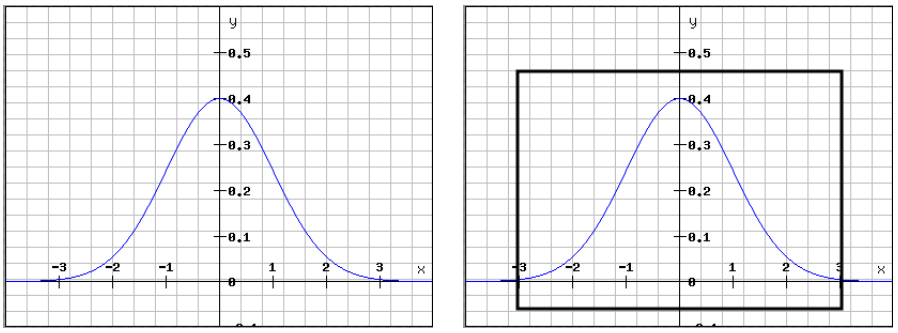


Figure 6.5: Number of positive and negative experiences with the service provider.

Belief recovery of the service provider after it fails to provide the best service for a while appears to be much slower than the initial belief growth for the service provider. Thus, initially when providing only the best service, the belief in the service provider grows from 0 to 0.55 after 25 interactions. However after failing to provide the best service for 14 interactions and then starting to provide the best service again, the belief in the service provider grows from 0.25 to 0.45 after 25 interactions of providing the best service. Likewise, uncertainty in the service provider decreases by 0.55 after 25 initial best service providing interactions. However after failure to provide the best service the uncertainty in the service provider decreases by 0.1 from 0.6 to 0.5 after 25 best service providing interactions.

6.3 Reputation evolution of different service providers

Here we illustrate the feature of comparing and analyzing reputation of different service providers of the same service that pursue different strategies. Let us assume there are two service providers in a multiagent system. The first service provider is the same as described above: agent 0 with the Weibull distribution as a service providing pattern. This service providing pattern means providing the best service with some periods of failing to provide the best service. Let us compare this pattern with a pattern that includes constant variation from providing the worst to providing the best service. This pattern can be described for example by the function of the standard normal distribution: $f(x) = \frac{1}{\sqrt{2\Pi}} \cdot e^{-\frac{x^2}{2}}$. Figure 6.6(a) represents the distribution function.



(a) Standard Normal distribution.

(b) Part of the Standard Normal distribution in use.

Figure 6.6: Service providing pattern.

Let us choose a piece of the distribution from $x = -3$ to $x = 3$ that will be in use as

a pattern for the service providing. Figure 6.6(b) highlights the selected part. Step for the service providing is 0.5.

To compare reputations of the service providers, the service providers should provide the same service and the requesting patterns for the services from both service providers should be the same. However if two agents provide the same service with the service providing patterns described above, only agent 0 will be asked for the service, as it starts providing the best service from the beginning and agents will give good referrals about agent 0 only. For this reason we will fake the existence of only one service in the multiagent system. Thus, we will initialize two identical services with different ids. Agent 0 is the only service provider for one of the services. Agent 1 is the only service provider for another service. Requesting patterns for both services are identical. In this case we can track the evolution of reputations of the service providers and compare them.

Let us take the time period for the comparison when the pattern of both of the patterns complete one round. The first pattern which includes Weibull distribution starts from $x = -0.6$ to $x = 2$ with the step 0.07. This means that the whole round will take around 37 interactions. The second pattern which includes Standard Normal distribution starts from $x = -3$ to $x = 3$ with the step 0.5. For this pattern the whole round will take 12 interactions. Thus, the minimum number of the traced interactions should be 37. Figure 6.7 represents belief values of the service providers' reputations. Figure 6.8 represents uncertainty values of the service providers' reputations. Figure 6.9 represents the number of positive experiences with the service providers. Figure 6.10 represents the number of negative experiences with the service providers.

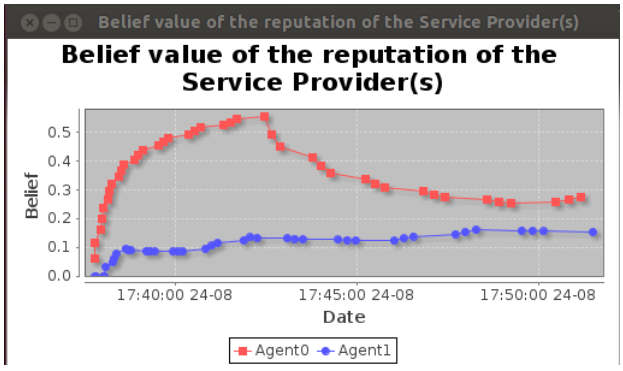


Figure 6.7: Belief values of the service providers' reputations.

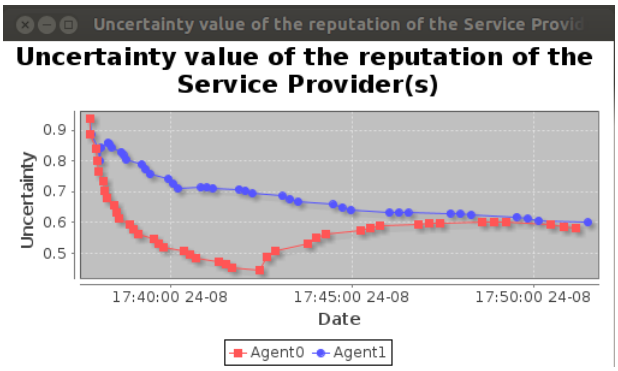


Figure 6.8: Uncertainty values of the service providers' reputations.

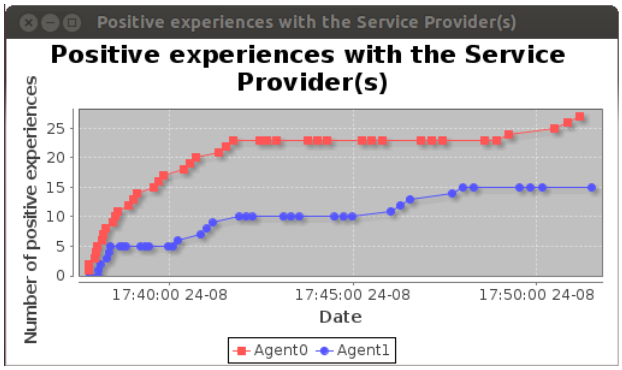


Figure 6.9: Number of positive experiences with the service providers.

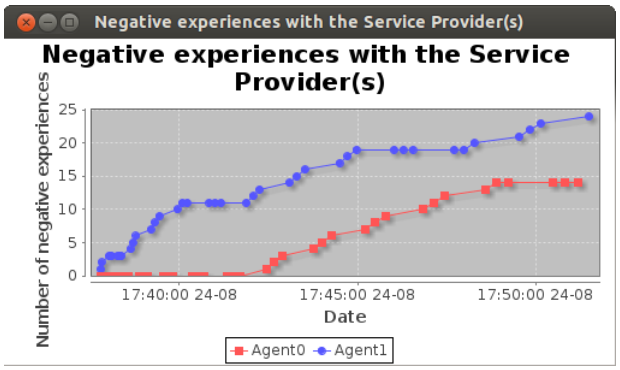


Figure 6.10: Number of negative experiences with the service providers.

There are approximately 40 traced interactions. For the agent 1 we can see that even it constantly changes the service providing pattern, the belief in it has an increasing

trend. It has increased from 0 to 0.15 after the traced interactions. For the agent 0 we can also observe increasing trend of the belief value in it. Thus, the belief value has increased from 0 to 0.29 after the same amount interactions. We can see that at any point the belief in the service provider 0 is higher than the belief in the service provider 1.

Uncertainty values for both service providers have a decreasing trend. For the agent 0 uncertainty value decreases from 1 to 0.59 after the traced interactions. For the agent 1 uncertainty value decreases from 1 to 0.6 after the same number of interactions. Thus, we can see that eventually the uncertainty values in the service providers are almost the same. However in total there were around 27 positive and 14 negative experiences with the agent 0, while 15 positive and 24 negative experiences with the agent 1.

7 Conclusion

This section concludes the results of the thesis work and outlines the future work.

Technology-isolated inter-enterprise collaboration is more beneficial compared to the conventional way of collaborating where enterprises are technology-dependent. However such technology-isolated collaboration increases the number of the potential partners and thus can lead to the market deterioration. For this reason it is crucial to create or configure a multiagent system for the agents' collaboration that can serve the guard purpose, supporting acceptable behavior from the agents and neglecting misbehavior among them.

There are many trust and reputation management algorithms that have been suggested in the research community for this purpose. These algorithms can differ in their assumptions, possible scenarios, reputation update policies and so on, however the main purpose stays the same. In the thesis we have come up with our general model of a multiagent system behavior.

The goal of the thesis was to implement a generic tool that can be used for evaluation and comparison of different trust or reputation management algorithms. To achieve this goal first we have studied the state of the art in the field of both trust and reputation management algorithms and existing tools for evaluating and comparing these algorithms. Analysis of the cutting-edge trust and reputation management algorithms allowed us to come up with our own generic model of a multiagent system behavior. Moreover, this analysis was crucial for making design decisions for our tool that must be generic enough to be able to embed as many different existing algorithms as possible. Analysis of the existing tools for evaluating and comparing trust and reputation management algorithms allowed us to come up with the design of the tool that overcomes existing tools' limitations.

Second, we have come up with the comprehensive list of the evaluation characteristics for the algorithms and identified the target characteristic that can be estimated using our system. Namely, the main characteristic that can be evaluated using our system is a service provider's reputation update policy. Future improvements of the system include implementation of all the other identified evaluation characteristics in the system. These characteristics include: referrer's reputation update policy, trust update policy, attack resistance of the algorithm, incentives of the algorithm, performance characteristics of the tool itself (e.g. feasibility, usability or scalability).

Third, we have implemented and described the tool for simulating a multiagent

system behavior. In the tool all the agents follow the same trust and reputation management algorithm. The system was meant to be generic enough to be able to plug in any trust and reputation management algorithm.

Fourth, we have described probabilistic approach for maintaining trust based on evidence according to our terms and above-presented vision. Moreover, we have embedded this algorithm into our tool and presented some implementation solutions.

Finally, we have illustrated evaluation of the service provider's reputation update policy of the probabilistic approach for maintaining trust based on evidence using our tool. To evaluate service provider's reputation update policy when the behavior of a service provider is normal and expected, we have illustrated the following aspects: agents' reactivity to changes in the behavior of a service provider and reputation evolution of different service providers of the same service that pursue pursue different strategies. To evaluate service provider's reputation update policy when the behavior of a service provider is normal however possibly assumed as misbehavior, we have illustrated the following aspect: recovery of a service provider's reputation if it fails to provide a service for a while.

To outline the future work for the current system we need to remember that the main purpose of the system is a complete behavior evaluation of a trust or reputation management algorithm and support for the algorithms' comparison. Our vision of the complete behavior evaluation of a trust or reputation management system is presented in Section 4.1. According to that vision future work for in the priority order includes: implementing support for the evaluation of the update reputation policy from a referrer's point of view; implementing support for the evaluation of the attack resistance of the system; implementing support for the evaluation of the incentives the algorithm creates; and implementing support for the evaluation of the system's feasibility, usability, scalability and other performance parameters.

Currently we have implemented the system's support for the evaluation of the update reputation policy from a service provider's point of view. This support includes design and implementation of the generic logic for the service providers' reputation update. Based on this defined generic logic different reputation management algorithms with their special features can be plugged into the system. Moreover, the tool provides a graphical representation of the reputation evolution of a single provider or a group of service providers on the same graph. However the current system is partly missing the support for evaluation of the update reputation policy from a referrer's point of view. The design part of the generic logic for the referrers'

reputation update is presented in the current paper, however the system lacks the implementation part and graphical representation of the results.

Support for the evaluation of the attack resistance of the system requires first of all identifying a comprehensive list of possible attacks from service provider's and referrer's sides. Attacks are based on the possible vulnerabilities of the systems. Section 3.2.3 presents known to us at the moment possible vulnerabilities of the systems from both service provider's and referrer's directions.

However the most important part of the evaluation of the attack resistance of the system lies in the design solution. How to evaluate an attack resistance of the system? What should be the criteria? Should the criteria be connected with the gains of the attackers or losses of the benevolent agents? How to evaluate the gains or losses? Should monetary relationships between agents be introduced to the environment for obtaining more representative results of the evaluations? These are just some questions that should be considered to find a solid solution for the evaluation of the attack resistance of the system.

Current research does not provide any good solution for the evaluation of the incentives the algorithm creates. One of such incentives can be for example rewarding participants more for many small transactions rather than for a few larger ones. Thus future work in this direction implies first researching the possible ways for the system's incentives evaluation and second coming up with a solid design and implementation solution.

Most of the performance parameters (e.g. scalability or usability) can be evaluated regardless of the system's implementation features. However the system can still provide some support. For example for the usability evaluation of the particular algorithm in the system, the tool could automatically calculate how many steps are required from the user to get a desired algorithm simulated. This could mean for example how many parameters of the algorithm must be specified in the config.properties file. The more parameters must be specified, the less usable the particular algorithm in the system appears to the user, as it will require from the user certain knowledge of the algorithm's logic.

References

Ake70 Akerlof, G. A., The market for 'Lemons': Quality uncertainty and the market mechanism. *The Quarterly Journal of Economics*, 84,3(1970), pages 488–500. URL `http://EconPapers.repec.org/RePEc:tpr:qjecon:v:84:y:1970:i:3:p:488-500`.

AbH00 Abdul-Rahman, A. and Hailes, S., Supporting trust in virtual communities. *Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 6 - Volume 6*, HICSS '00, Washington, DC, USA, 2000, IEEE Computer Society, page 6007, URL `http://dl.acm.org/citation.cfm?id=820262.820322`.

ART13 The agent reputation and trust (ART) testbed. October 2013, URL `http://megatron.iiia.csic.es/art-testbed/`. [01.10.2013].

ChE11 Chandrasekaran, P. and Esfandiari, B., A model for a testbed for evaluating reputation systems. *Proceedings of the 2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, Changsha, Hunan Province P.R. China, November 2011, pages 296–303, URL `http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6120832`.

CGS03 Cahill, V., Gray, E., Seigneur, J.-M., Jensen, C. D., Chen, Y. and et. al, Using trust for secure collaboration in uncertain environments. *IEEE Pervasive Computing*, 2,3(2003), pages 52–61. URL `http://dx.doi.org/10.1109/MPRV.2003.1228527`.

Cha05 Chadwick, D. W., Operational models for reputation servers. *iTrust*, volume 3477 of *Lecture Notes in Computer Science*. Springer, 2005, pages 108–115, URL `http://dx.doi.org/10.1007/11429760_8`.

CNS03 Carbone, M., Nielsen, M. and Sassone, V., A formal model for trust in dynamic networks. *In proc. of International Conference on Software Engineering and Formal Methods (SEFMâ03)*. Society Press, 2003, pages 54–63.

DeA04 Despotovic, Z. and Aberer, K., Maximum likelihood estimation of peers' performance in P2P networks. *The Second Workshop on the Economics of Peer-to-Peer Systems*, 2004.

FKM06 Fullam, K., Klos, T., Muller, G., Sabater, J., Schlosser, A., Topol, Z., Barber, S., Rosenschein, J., Vercouter, L. and Voss, M., The agent reputation and trust (ART) testbed game description (version 2.1). Technical Report, 2006. URL http://megatron.iiia.csic.es/art-testbed/pdf/SpecSummary.pdf.

FKO04 Fernandes, A., Kotsovinos, E., Ostring, S. and Dragovic, B., Pinocchio: Incentives for honest participation in distributed trust management. In *Trust Management*, Jensen, C., Poslad, S. and Dimitrakos, T., editors, volume 2995 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2004, pages 63–77, URL http://dx.doi.org/10.1007/978-3-540-24747-0_6.

Gal11 Gal-Oz, N., Models for trust-based reputation in and across virtual communities. Doctoral thesis, Ben-Gurion University of the Negev, Department of Computer Science, Beer-Sheva, March 2011.

GGH08 Gal-Oz, N., Gudes, E. and Hendler, D., A robust and knot-aware trust-based reputation model. In *Trust Management II*, volume 263 of *The International Federation for Information Processing*, Springer US, 2008, pages 167–182, URL http://dx.doi.org/10.1007/978-0-387-09428-1_11.

HJS06 Huynh, T. D., Jennings, N. R. and Shadbolt, N. R., An integrated trust and reputation model for open multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 13,2(2006), pages 119–154. URL http://dx.doi.org/10.1007/s10458-005-6825-4.

IBJ03 Ismail, R., Boyd, C., Jøsang, A. and Russel, S., Strong privacy in reputation systems. *Proceedings of the 4th international Workshop on Information Security Applications (WISA)*, August 2003.

JuF03 Jurca, R. and Faltings, B., An incentive compatible reputation mechanism. *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, AAMAS '03, New York, NY, USA, 2003, ACM, pages 1026–1027, URL http://doi.acm.org/10.1145/860575.860778.

JøH07 Jøsang, A. and Haller, J., Dirichlet reputation systems. *Proceedings of the The Second International Conference on Availability, Reliability and*

Security, ARES'07, Washington, DC, USA, 2007, IEEE Computer Society, pages 112–119, URL http://dx.doi.org/10.1109/ARES.2007.71.

JøI02 Jøsang, A. and Ismail, R., The Beta reputation system. *In Proceedings of the 15th Bled Electronic Commerce Conference*, 2002, pages 324–337.

JIB07 Jøsang, A., Ismail, R. and Boyd, C., A survey of trust and reputation systems for online service provision. *Decis. Support Syst.*, 43,2(2007), pages 618–644. URL http://dx.doi.org/10.1016/j.dss.2005.05.019.

Jøs01 Jøsang, A., A logic for uncertain probabilities. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 9,3(2001), pages 279–311. URL http://dl.acm.org/citation.cfm?id=565980.565981.

KBR05 Kinateder, M., Baschny, E. and Rothermel, K., Towards a generic trust model – comparison of various trust update algorithms. *Trust Management*, volume 3477 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2005, pages 119–134, URL http://dx.doi.org/10.1007/11429760_13.

KeC06 Kerr, R. and Cohen, R., Modeling trust using transactional, numerical units. *Proceedings of the 2006 International Conference on Privacy, Security and Trust: Bridge the Gap Between PST Technologies and Business Services*, PST '06, New York, NY, USA, 2006, ACM, pages 21:1–21:11, URL http://doi.acm.org/10.1145/1501434.1501460.

KeC09 Kerr, R. and Cohen, R., Smart cheaters do prosper: defeating trust and reputation systems. *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '09, Richland, SC, 2009, International Foundation for Autonomous Agents and Multiagent Systems, pages 993–1000, URL http://dl.acm.org/citation.cfm?id=1558109.1558151.

KeC10 Kerr, R. and Cohen, R., TREET: the Trust and Reputation Experimentation and Evaluation Testbed. *Electronic Commerce Research*, volume 10. Springer Netherlands, 2010, pages 271–290, URL http://dx.doi.org/10.1007/s10660-010-9056-y.

KGB09 Khosravifar, B., Gomrokchi, M., Bentahar, J. and Thiran, P., Maintenance-based trust for multi-agent systems. *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '09, Richland, SC, 2009, International Foundation for Autonomous Agents and Multiagent Systems, pages 1017–1024, URL `http://dl.acm.org/citation.cfm?id=1558109.1558154`.

KRR08 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`.

KSG03 Kamvar, S. D., Schlosser, M. T. and Garcia-Molina, H., The Eigentrust algorithm for reputation management in P2P networks. *Proceedings of the 12th international conference on World Wide Web*, WWW '03, New York, NY, USA, 2003, ACM, pages 640–651, URL `http://doi.acm.org/10.1145/775152.775242`.

Lev09 Levien, R., Attack-resistant trust metrics. *Computing with social trust*. Springer, 2009, pages 121–132.

Man98 Manchala, D. W., Trust metrics, models and protocols for electronic commerce transactions. *Proceedings of the The 18th International Conference on Distributed Computing Systems*, ICDCS '98, Washington, DC, USA, 1998, IEEE Computer Society, pages 312–, URL `http://dl.acm.org/citation.cfm?id=850926.851678`.

MMH02 Mui, L., Mohtashemi, M. and Halberstadt, A., A computational model of trust and reputation for e-businesses. *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02) - Volume 7*, Washington, DC, USA, 2002, IEEE Computer Society, pages 188–, URL `http://dl.acm.org/citation.cfm?id=820745.821158`.

MRZ02 Miller, N., Resnick, P. and Zeckhauser, R., Eliciting honest feedback in electronic markets. Working paper series, Harvard University, John F. Kennedy School of Government, 2002. URL `http://ideas.repec.org/p/ecl/harjfk/rwp02-039.html`.

PDS09 Paradesi, S., Doshi, P. and Swaika, S., Integrating behavioral trust in web service compositions. *Proceedings of the 2009 IEEE International Conference on Web Services*, ICWS '09, Washington, DC, USA, 2009, IEEE Computer Society, pages 453–460, URL `http://dx.doi.org/10.1109/ICWS.2009.106`.

PPK05 Papagelis, M., Plexousakis, D. and Kutsuras, T., Alleviating the sparsity problem of collaborative filtering using trust inferences. In *Trust Management*, volume 3477 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2005, pages 224–239, URL `http://dx.doi.org/10.1007/11429760_16`.

RHK11 Ruohomaa, S., Hankalahti, A. and Kutvonen, L., Detecting and reacting to changes in reputation flows. In *Trust Management V*, volume 358 of *IFIP Advances in Information and Communication Technology*, Springer Berlin Heidelberg, 2011, pages 19–34, URL `http://dx.doi.org/10.1007/978-3-642-22200-9_5`.

RuK05 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`.

RuK08 Ruohomaa, S. and Kutvonen, L., Making multi-dimensional trust decisions on inter-enterprise collaborations. *Seventh International Conference on Availability, Reliability and Security*, pages 873–880. URL `http://doi.ieeecomputersociety.org/10.1109/ARES.2008.32`.

RuK13 Ruohomaa, S. and Kutvonen, L., Behavioural evaluation of reputation-based trust systems. In *Enterprise Interoperability*, volume 144 of *Lecture Notes in Business Information Processing*, Springer Berlin Heidelberg, 2013, pages 158–171, URL `http://dx.doi.org/10.1007/978-3-642-36796-0_14`.

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`.

RKK12 Ruohomaa, S., Kaur, P. and Kutvonen, L., From subjective reputation to verifiable experiences augmenting peer-control mechanisms for open service ecosystems. In *Trust Management VI*, volume 374 of *IFIP Advances in Information and Communication Technology*, Springer Berlin Heidelberg, 2012, pages 142–157, URL `http://dx.doi.org/10.1007/978-3-642-29852-3_10`.

RKZ00 Resnick, P., Kuwabara, K., Zeckhauser, R. and Friedman, E., Reputation systems. *Commun. ACM*, 43,12(2000), pages 45–48. URL `http://doi.acm.org/10.1145/355112.355122`.

RRR07 Reece, S., Roberts, S., Rogers, A. and Jennings, N. R., A multi-dimensional trust model for heterogeneous contract observations. *Proceedings of the 22nd national conference on Artificial intelligence - Volume 1*, AAAI'07. AAAI Press, 2007, pages 128–135, URL `http://dl.acm.org/citation.cfm?id=1619645.1619666`.

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.

Ruo12 Ruohomaa, S., The effect of reputation on trust decisions in inter-enterprise collaborations. Doctoral thesis, University of Helsinki, Department of Computer Science, Helsinki, May 2012.

ReZ02 Resnick, P. and Zeckhauser, R., Trust among strangers in internet transactions: Empirical analysis of ebay's reputation system. In *The Economics of the Internet and E-Commerce*, volume 11 of *Advances in Applied Microeconomics*, Elsevier Science, 2002, pages 127–157, URL `http://www.si.umich.edu/~presnick/papers/ebayNBER/index.html`.

SBL04 Staab, S., Bhargava, B., Lilien, L., Rosenthal, A. and et. all, The pudding of trust. *IEEE Intelligent Systems*, 19,5(2004), pages 74–88. URL `http://dx.doi.org/10.1109/MIS.2004.52`.

Sin03 Singh, M., Trustworthy service composition: challenges and research questions. *Trust, Reputation, and Security: Theories and Practice*, volume 2631 of Lecture Notes in Computer Science. Springer Berlin Heidel-

berg, 2003, pages 39–52, URL http://www.csc.ncsu.edu/faculty/mpsingh/papers/mas/trust-deception-02.pdf.

SaS02 Sabater, J. and Sierra, C., Reputation and social network analysis in multi-agent systems. *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, AAMAS '02, New York, NY, USA, 2002, ACM, pages 475–482, URL http://doi.acm.org/10.1145/544741.544854.

SaS05 Sabater, J. and Sierra, C., Review on computational trust and reputation models. *Artif. Intell. Rev.*, 24,1(2005), pages 33–60. URL http://dx.doi.org/10.1007/s10462-004-0041-5.

TPJ06 Teacy, W. T. L., Patel, J., Jennings, N. R. and Luck, M., TRAVOS: Trust and Reputation in the Context of Inaccurate Information Sources. *Autonomous Agents and Multi-Agent Systems*, 12, pages 183–198. URL http://dx.doi.org/10.1007/s10458-006-5952-x.

WHS11 Wang, Y., Hang, C.-W. and Singh, M., A probabilistic approach for maintaining trust based on evidence. *Journal of Artificial Intelligence Research*, 40,1(2011), pages 221–267. URL http://www.jair.org/media/3108/live-3108-5411-jair.pdf.

WaS06 Wang, Y. and Singh, M., Trust representation and aggregation in a distributed agent system. *Proceedings of the 21st National Conference on Atrificial Intelligence (AAAI)*, Boston, MA, USA, 2006, AAAI Press, pages 1425–1430.

WaS07 Wang, Y. and Singh, M., Formal trust model for multiagent systems. *Proceedings of the 20th International Joint Conference on Atrificial Intelligence (IJCAI)*, Hyderabad, India, 2007, IJCAI, pages 1551–1556.

WaS10 Wang, Y., and Singh, M., Evidence-based trust: A mathematical model geared for multiagent systems. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 5,4(2010), pages 14:1–14:28.

XiL04 Xiong, L. and Liu, L., PeerTrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE Trans. on Knowl. and Data Eng.*, 16,7(2004), pages 843–857. URL http://dx.doi.org/10.1109/TKDE.2004.1318566.

YuS02 Yu, B. and Singh, M. P., An evidential model of distributed reputation management. *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, AAMAS '02, New York, NY, USA, 2002, ACM, pages 294–301, URL `http://doi.acm.org/10.1145/544741.544809`.

ZiL04 Ziegler, C.-N. and Lausen, G., Spreading activation models for trust propagation. *Proceedings of the 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'04)*, Washington, DC, USA, 2004, IEEE Computer Society, pages 83–97, URL `http://dl.acm.org/citation.cfm?id=987681.987786`.

Appendix A. Representation of the probabilistic approach in our system

This Appendix shows some implementation decisions of the probabilistic approach for maintaining trust based on evidence [WHS11] in our system. Table A.1 describes main classes created for the algorithm implementation. Table A.2 presents main methods of the API for the algorithm.

Table A.1: Main classes for the probabilistic algorithm.

Class name	Class description	Field type	Field name	Field description
RepBelief	represents reputation value in the belief space	double	**belief**	belief value (b)
		double	**uncertainty**	uncertainty value (u)
RepEvidence	represents reputation value in the evidence space	double	**positiveExperiences**	represents positive experiences (r)
		double	**negativeExperiences**	represents negative experiences (s)
ServiceProviderRep_PA	represents service provider's reputation (PA stands for Probabilistic Algorithm)	RepEvidence	**evidenceRep**	service provider's reputation in the evidence space
		RepBelief	**beliefRep**	service provider's reputation in the evidence space
ReferrerRep_PA	represents referrer's reputation	RepEvidence	**evidenceRep**	referrer's reputation in the evidence space
		RepBelief	**beliefRep**	referrer's reputation in the evidence space

Table A.2: Main methods of the API for the probabilistic algorithm.

Owner class	Method name	Method description, return value	Parameter	Parameter description
RepBelief	**fromBeliefToEvidence**	transfers reputation representation from the belief to the evidence space, RepEvidence	double t_max	possible maximal number of experiences between two agents
			double e	precision of the calculations
RepEvidence	**fromEvidenceToBelief**	transfers reputation representation from the evidence to the belief space, void	RepBelief result	the result object the value of which will be updated
Service Provider-Rep_PA	**update Reputation**	updates reputation value for the service provider, void	double newPositiveExperience	value of the obtained positive experience
			double newNegativeExperience	value of the obtained negative experience
Service Provider-Rep_PA	**concatenatedRep**	concatenates different reputation values, ServiceProviderRep_PA	RepBelief TR_in_RF	reputation value in the belief space that trustor places in a referrer
			RepBelief RF_in_SP	reputation value in the belief space that referrer places in a service provider
Service Provider-Rep_PA	**aggregatedRep**	aggregates different reputation values, ServiceProviderRep_PA	RepBelief[] TR_in_RF	an array of reputation values in the belief space that N trustors place in a referrer

Owner class	Method name	Method description, return value	Parameter	Parameter description
			RepBelief[] RF_in_SP	an array of reputation values in the belief space that referrers place in a service provider
ReferrerRep_PA	**updateReputation**	updates reputation value for the referrer, void	RepEvidence SP_prev_rep	reputation value of the service provider in the evidence space before the new experience
			RepEvidence SP_new_rep	reputation value of the service provider in the evidence space after the experience
			RepEvidence Ref_prev_rep	reputation value of the referrer in the evidence space before the experience
			double beta	discounting factor

Appendix B. Agents' reactivity to changes in the behavior of a service provider

This Appendix shows some simulation results for a multiagent system that is configured according to the Section 6.1. The main purpose of the simulation run is to illustrate system's features for tracking agents' reactivity to changes in the behavior of a service provider. Listing B.1 presents system's initialization. The format of the further listings is shortened for the compacting purposes and to point out the main places of interest. Listing B.2 presents initial service providing pattern with the best service providing characteristics. Listing B.3 presents changed behavior of the service provider.

Listing B.1: Initialization of the multiagent system

```
21-Aug-2013 19:47:21 INFO  SimulationManager-68
Initializing a multiagent system...
21-Aug-2013 19:47:22 INFO  SimulationManager-71 Initializing patterns...
21-Aug-2013 19:47:22 INFO  Initializer-63
Pattern index: 0, pattern name: Weibull curve, function:
Math.exp(-Math.pow(x,50)); xMin = -0.6; xMax = 2.0;
xCurrent = -0.6; step = 0.07
21-Aug-2013 19:47:22 INFO  Initializer-63
Pattern index: 1, pattern name: Vertical line, function:
3; xMin = 3.0; xMax = 3.0; xCurrent = 3.0; step = 0.0
21-Aug-2013 19:47:22 INFO  SimulationManager-73
...Initializing patterns [OK]
21-Aug-2013 19:47:22 INFO  SimulationManager-74 Initializing services...
21-Aug-2013 19:47:22 INFO  Initializer-79
ServiceId: 0, domain: IT, type: Printing
21-Aug-2013 19:47:22 INFO  Initializer-86
ServiceId: 0, dimension: Quality of paper
21-Aug-2013 19:47:22 INFO  Initializer-100
ServiceId: 0, values: [Bad, Okay, Good, Perfect]
21-Aug-2013 19:47:22 INFO  Initializer-150
Service id: 0, requesting pattern: Vertical line
21-Aug-2013 19:47:22 INFO  SimulationManager-76
...Initializing services [OK]
21-Aug-2013 19:47:22 INFO  SimulationManager-77 Initializing agents...
21-Aug-2013 19:47:22 INFO  Initializer-310
Agent 0 [ serviceId = 0, service domain: IT, service type:
Printing, service providing pattern: Weibull curve; ]
21-Aug-2013 19:47:22 INFO  Initializer-301
```

```
Agent 1 [ service domain is empty ]
21-Aug-2013 19:47:22 INFO  Initializer-301
Agent 2 [ service domain is empty ]
...
21-Aug-2013 19:47:22 INFO  Initializer-301
Agent 998 [ service domain is empty ]
21-Aug-2013 19:47:22 INFO  Initializer-301
Agent 999 [ service domain is empty ]
21-Aug-2013 19:47:22 INFO  Initializer-313
Total number of service instances: 1
21-Aug-2013 19:47:22 INFO  SimulationManager-79
...Initializing agents [OK]
21-Aug-2013 19:47:22 INFO  SimulationManager-80
Preparing new reports...
21-Aug-2013 19:47:31 INFO  SimulationManager-82
...Preparting new reports [OK]
21-Aug-2013 19:47:31 INFO  SimulationManager-83
...Initializing a multiaget system [OK]
```

Listing B.2: Initial service providing pattern

```
19:47:31
Requesting the services. Every 10 second(s)...
Requesting ServiceId = 0, domain = IT, type = Printing.
Number of requesting agents = 3
19:47:32
Number of service providers: 1 for the serviceId: 0.
Requesting agent is a service provider itself: false
Agent 500 collected referrals for 0 service provider(s)
Selecting service provider based on local information.
Agent 500 requests service (id = 0) from the Agent 0...
Provided service: Dimension = Quality of paper, value = Perfect.
Service provider's␣0␣reputation␣before␣update:
Number␣of␣positive␣experiences:␣0.0,␣number␣of␣negative␣experiences:␣0.0.
Belief:␣0.0,␣uncertainty:␣1.0
Update␣of␣reputation␣for␣a␣service␣provider,
agentId:␣500,␣serviceProviderId:␣0
Service␣provider's 0 reputation after update:
Number of positive experiences: 1.0, number of negative experiences: 0.0.
Belief: 0.25, uncertainty: 0.75
...
19:48:47
Number of service providers: 1 for the serviceId: 0.
Requesting agent is a service provider itself: false
For the service provider id: 0 number of referrals: 18.
```

```
Agent 449 collected referrals for 1 service provider(s)
Concatenating 18 experience(s) for the service provider id = 0
19:48:54
Aggregating 18 experiences for the service provider id = 0
Selecting service provider based on gathered information.
Agent 449 requests service (id = 0) from the Agent 0...
Provided service: Dimension = Quality of paper, value = Perfect.
Service provider's␣0␣reputation␣before␣update:
Number␣of␣positive␣experiences:␣0.0,␣number␣of␣negative␣experiences:␣0.0.
Belief:␣0.0,␣uncertainty:␣1.0
Updating␣reports␣for␣the␣service␣provider(s)␣reputations
Update␣of␣reputation␣for␣a␣service␣provider,
agentId:␣449,␣serviceProviderId:␣0
Service␣provider's 0 reputation after update:
Number of positive experiences: 1.0, number of negative experiences: 0.0.
Belief: 0.25, uncertainty: 0.75
```

Listing B.3: Changed behavior of the service provider

```
19:48:54
Number of service providers: 1 for the serviceId: 0.
Requesting agent is a service provider itself: false
For the service provider id: 0 number of referrals: 19
Agent 158 collected referrals for 1 service provider(s)
Concatenating 19 experience(s) for the service provider id = 0
19:49:02
Aggregating 19 experiences for the service provider id = 0.
Selecting service provider based on gathered information...
Agent 158 requests service (id = 0) from the Agent 0...
Provided service: Dimension = Quality of paper, value = Good
Service provider's␣0␣reputation␣before␣update:
Number␣of␣positive␣experiences:␣0.0,␣number␣of␣negative␣experiences:␣0.0.
Belief:␣0.0,␣uncertainty:␣1.0
Updating␣reports␣for␣the␣service␣provider(s)␣reputations
Update␣of␣reputation␣for␣a␣service␣provider,
agentId:␣158,␣serviceProviderId:␣0
Service␣provider's 0 reputation after update:
Number of positive experiences: 1.0, number of negative experiences: 0.0.
Belief: 0.25, uncertainty: 0.75
...
19:49:35
Number of service providers: 1 for the serviceId: 0.
Requesting agent is a service provider itself: false
For the service provider id: 0 number of referrals: 24
Agent 662 collected referrals for 1 service provider(s)
```

```
Concatenating 24 experience(s) for the service provider id = 0
19:49:45
Aggregating 24 experiences for the service provider id = 0
Selecting service provider based on gathered information...
Agent 662 requests service (id = 0) from the Agent 0...
Provided service: Dimension = Quality of paper, value = Okay
Service provider's␣0␣reputation␣before␣update:
Number␣of␣positive␣experiences:␣0.0,␣number␣of␣negative␣experiences:␣0.0.
Belief:␣0.0,␣uncertainty:␣1.0
Updating␣reports␣for␣the␣service␣provider(s)␣reputations
Update␣of␣reputation␣for␣a␣service␣provider,
agentId:␣662,␣serviceProviderId:␣0
Service␣provider's 0 reputation after update:
Number of positive experiences: 0.0, number of negative experiences: 1.0.
Belief: 0.0, uncertainty: 0.75
...
```