Neural Trust Model for Multi-agent Systems

Gehao Lu

School of Computing and Engineering

University of Huddersfield

A thesis submitted for the degree of

Doctor of Philosophy

2011 Aug

Abstract

Introducing trust and reputation into multi-agent systems can significantly improve the quality and efficiency of the systems. The computational trust and reputation also creates an environment of survival of the fittest to help agents recognize and eliminate malevolent agents in the virtual society. The thesis redefines the computational trust and analyzes its features from different aspects. A systematic model called Neural Trust Model for Multi-agent Systems is proposed to support trust learning, trust estimating, reputation generation, and reputation propagation. In this model, the thesis innovates the traditional Self Organizing Map (SOM) and creates a SOM based Trust Learning (STL) algorithm and SOM based Trust Estimation (STE) algorithm. The STL algorithm solves the problem of learning trust from agents' past interactions and the STE solve the problem of estimating the trustworthiness with the help of the previous patterns. The thesis also proposes a multi-agent reputation mechanism for generating and propagating the reputations. The mechanism exploits the patterns learned from STL algorithm and generates the reputation of the specific agent. Three propagation methods are also designed as part of the mechanism to guide path selection of the reputation. For evaluation, the thesis designs and implements a test bed to evaluate the model in a simulated electronic commerce scenario. The proposed model is compared with a traditional arithmetic based trust model and it is also compared to itself in situations where there is no reputation mechanism. The results state that the model can significantly improve the quality and efficacy of the test bed based scenario. Some design considerations and rationale behind the algorithms are also discussed based on the results.

Acknowledgements

I would like to express my gratitude to all those who gave me the possibility to complete this thesis. I am deeply indebted to my supervisor Prof. Joan Lu from University of Huddersfield whose help, stimulating suggestions and encouragement helped me in all the time of research for and writing of this thesis.

I want to thank the School of Software of Yunnan University for giving me permission to commence this thesis in the first instance, to do the necessary research work and to use departmental data. I have furthermore to thank the Dean of the School of Software, Prof. Tong Li and the Secretary of the School of Software, Prof. Shipu Wang who gave and confirmed this permission and encouraged me to go ahead with my thesis.

My colleagues from the School of Software supported me in my research work. I want to thank them for all their help, support, interest and valuable hints. Especially I am obliged to Professor Shaowen Yao, who gave me important early financial support on my degree and research. I also want to thank Mr.Hao Li for all his assistance on the the Lab facilities and computers. My friend, Tony Chen looked closely at the final version of the thesis for English style and grammar, correcting both and offering suggestions for improvement.

Especially, I would like to give my special thanks to my wife Nan Yu, my Father Biao Lu, and my Mother Moshu Luo whose patient love enabled me to complete this work.

Contents

List of Figures

List of Tables

1

Introduction

1.1 Introduction

The characteristic of "intelligence" is usually attributed to humans. Grant more intelligence to machines has always been a fascinating dream in fictional movies, however, it is an important research direction for computer scientists as well. The traditional symbol based artificial intelligence is confined to an individual program or an application in the form of strict logic representation. The distributed artificial intelligence focuses on sociality of the systems and the computing intelligence emphasizes the emergent properties of the intelligence. A multi-agent system is an attempt to create intelligent applications that exploit the advantages from these different types of intelligences. It is an aggregate or even a society of intelligent agents, at the same time, those agents possesses the abilities that different intelligent theories bestow. The coordination and cooperation is the main research subject in multi-agent systems. Inside these two relationships, the idea of intelligent agents capable of trusting each other and be able to build a reputation network analogous to human society is an interesting and important question which is deserved to research. Such trust and reputation is especially designed for computer programs. It can be named as computational trust and computational reputation, or more specifically, it can be called as agent trust and agent reputation in multi-agent systems. Adding computational trust into the multi-agent system can bring many benefits. The efficiency of the interactions is greatly enhanced because the agents no longer need to waste time on negotiation with other agents due to lack of trust, and the agents no longer need to interact with agents with bad reputations.

The existence of trust and reputation forms an environment that pursues the rule of survival of the fittest. If an agent loses trusts or even reputation, the chance of further interactions are reduced or removed. Thus, the quality of the interaction is also greatly improved because of the elimination of the bad agents.

1.2 Importance and significance

There many mechanisms and algorithms that have been proposed for determining the computational trust and the computational reputation. Some of them have already been put into practice in industries such as eBay's reputation system (Resnick et al., 2006), while some are still research topics or research results such as (Zacharia et al., 1999), Marsh's Model (Marsh, 1994*b*), RegreT (Sabater and Sierra, 2002), Referral Reputation (Yu and Singh, 2000), FIRE (Huynh et al., 2004) and TRAVOS (Teacy et al., 2006). Most models are mathematical models that are based on summation or product of different dimensions with selected weights representing their influences. Some models (Teacy et al., 2006) are based on probability theory or statistical methods while most models presume the semantics behind trust is consistent to all the agents.

Although the research on trust and reputation have achieved great process, there are some still some problems that need to be solved:

- Lack of adaptability: Most models (Marsh, 1994*b*; Resnick et al., 2006; Sabater and Sierra, 2002; Zacharia et al., 1999)are based on delicate arithmetic equations or probabilistic statistical methodologies such as the Bayesian system. These models may work fine in specific domains and specific scenarios but once the domains are changed or scenarios are changed, there is no way to adjust the equation of calculating trust. The rigidity of the mathematical models prevents them from adapting to the ever-changing environments. Unluckily, the environments for most multi-agent system are complex and dynamic. Autonomous agents are not simple task assigned programs. They have to be proactive and reactive in a complex environment. Thus they need the ability to learn from outside environments, learn from their interactions, and learn from the other agents.

- Lack of systematic approach: Some models (Marsh, 1994*b*) separate the trust and reputation as two different areas. They either carry out study simply on trust

or simply on reputation. But trust and reputation are two facets of a systematic whole. Trust is a facet viewed from the individual level and reputation is the other facet viewed from the social level. Reputation is the generated product of trust and trust spreads itself through reputation. Any solution which simply focuses on trust or reputation are not a systematic or complete solution.

- Lack of object semantics: Some models (Marsh, 1994*b*; Resnick et al., 2006; Zacharia et al., 1999) assume that the trust is put in a single background, thus they do not care what the difference of the backgrounds might lead to. In fact, most of the time, the backgrounds are different. In different backgrounds, or say "domains", the object of trust is various so that semantics of trust relationship is also variable. The scarcity of the research into concepts like domain, object, semantics may impede the multi-agent from forming federated societies which cross many domains.

- Lack of difference between trust and trustworthiness: Some models (Huynh et al., 2004; Marsh, 1994*b*; Resnick et al., 2006; Teacy et al., 2006; Yu and Singh, 2000) do not differentiate trust and trustworthiness, they usually use a numeric value to represent both of them. Such simplification works in simple arithmetic model and covers the nature of learning of autonomous agents. Take human trust model as a comparison, trust is a recognition process and trustworthiness is a decision making process. An autonomous agent should also have such two level models.

Through the analysis against these problems, the thesis builds a systematical model for trust and reputation. This model is named as neural trust model for multi-agent system which combines the techniques from both distributed artificial intelligence and computing intelligence. The significances of the model are described as follows:

- The neural trust model proposes that the recognition of trust is actually a problem of machine learning and the model is designed and implemented based on this viewpoint. This gives the multi-agent trust system the ability to adapt to the environment. This also makes the detailed information of interactions become the foundation of trust learning.

- The neural trust model is a systematic and complete model which contains learning of trust, estimation of trust, reputation generation and reputation propagation. In this model, reputation is not separated from trust; it is the calculated result of trust. The reputation is based on the estimation against specific agent. The computational trust is transformed to reputation to propagate on the network.

- The neural trust model can be applied in different domains instead of single domain. Single domain can only represent one set of related concepts. In order to represent various sets of concepts in the real world, it is necessary to use multi-dimensional trust to cope with the varieties of domains. In each domain, the trust learning is against a few specific dimensions so that the change of domain only changes the dimensions of learning algorithm.

- The neural trust model presents a two level trust mechanism: one level for learning trust from interactions and reputations, the other level for estimation and decision making. The model explicitly separates these two processes so that each process can be independently executed. This means an agent can also do these two jobs concurrently instead of sequentially execution.

1.3 Aims and tasks

The aim of the research is to propose a computational trust model for multi-agent systems. The model should be capable of learning trust, estimating trust, generating reputation, and propagating reputation. In order to realize this aim, several tasks must be done. (1) Define the concept of trust and reputation and analyze the influential factors of the concept; (2) Design a model framework that specifies the components and the relationship among components. Design the process how the model should work; (3) Propose the algorithm for learning trust and estimating trust, also find out the solution of using trust learned as trustworthiness decision making criteria. (4) Design the mechanism of reputation generation and reputation propagation. The generation should be based on results of trust learning. The propagation should specify the process and paths. (5) A test bed should be designed to evaluate the proposed model, algorithms and mechanisms in the above tasks.

1.4 Contributions

The thesis analyzes the idea of learning trust, proposes a systematical model for trust and reputation, proposes the STL algorithm and STE algorithm, designs the reputation generation and propagation mechanism, and implement a test bed. The thesis believes that the computational trust is a combinatory problem of machine learning and decision making. The form of representation is patterns that are emerged from information collected during agent interactions. The thesis also points out that the trustworthiness and untrustworthiness are actually decision making problems. Most of the traditional computational trust models are arithmetical or statistical models which deem the trust and reputation as a numeric value generated from delicate equations. The thesis proposes a trust and reputation model named "Neural trust learning and estimation model". This model establishes a complete framework to accomplish trust related activities. It depicts the relationship among algorithms and mechanisms. It defines the processes of trust learning, estimating and reputation generating, propagation.Two algorithms are proposed. One is SOM based Trust Learning algorithm and the other is SOM based Trust Estimation algorithm. Both algorithms modify the original Self Organized Map to produce patterns suitable for representing trust. The estimation algorithm helps agents make trustworthiness decision according to its memorized patterns. The thesis proposed a reputation propagation mechanism to support the generation and propagation of reputations. The mechanism designs three types of reputation propagation paths: Point-to-point based inquiry, broadcasting based propagation and observer. The reputation is produced as the difference between macro pattern and agent pattern. An equation for calculating the reputations is proposed and issues such as decaying along the propagation are also modeled in it. In order to evaluate the model, a computational trust test bed is designed and implemented. The test bed is built upon Java Agent Development Environment platform and it constructs a simulated electronic market which contains many autonomous agents divided into different roles. The test bed not only can be used in this thesis related experiment, but it also can be used to test the other types of agent trust models.

1.5 Application domains

The neural trust model is initially designed for multi-agent systems. In many scenarios of using autonomous agents, the model can be applied to improve qualities and efficiencies. In automated electronic commerce, the model can be used as a commercial reputation system. Agents receives transactional task and automatically execute the transaction in a virtual market. They evaluate the trust toward each other and finally form a reputational network. In computer games, the NPC (Non Player Character) can make use of the proposed model to learn trust from the interaction between NPCs or between NPC and human players. In agent based planning, the agents can find trustworthy partner agents to form an optimized plan based on the proposed model. In case there are interactions, the model can be exploited to some extent to help lubricate the virtual groups or virtual societies.

The application of the model is not limited to the field of multi-agent systems. It can also be used in regular computing tasks. In network security, trust is deemed as soft security (Barber and Kim, 2003) which restricts the relationship between programs or components. The proposed model can be transformed against trust of programs or trust of components instead of agents. The model can also be applied to the program that needs trust or reputation mechanisms.

The thesis is organized as eight chapters. Chapter 1 is the introduction, Chapter 2 and Chapter 3 reviews the background knowledge from multi-agent researches and trust reputation researches respectively. Chapter 4 to Chapter 7 is the major technical parts of the thesis. Chapter 4 describes the neural trust model in general and presents the framework that the remaining components work. Chapter 5 explains the STL algorithm and STE algorithm used for trust learning and estimation. Chapter 6 introduces the reputation generation mechanism and reputation propagation mechanism. Chapter 7 exhibits the results from testing and evaluation; the test bed design is also introduced in this chapter. Chapter 8 is the conclusion and future work.

2

Background Review

2.1 Introduction

Multi-agent systems are initially introduced as a branch of distributed artificial intelligence. In 1986, Minsky proposed the concept of agents and he thought some problems could be solved through the negotiation among the social individuals (Minsky and Murata, 2004). These social individuals are agents. Agents should be highly interactive and intelligent. Hewitt thought that it is difficult to define agent as it is to define intelligence (Hewitt, 1985). Wooldrage and Jennings thought that agents should be autonomous, socially interactive, proactive, and reactive (Wooldridge, 2002). Russell thought that agents can react through sensing the outside environments (E. and Yuhui, 2007).

With the increased size and complexity of the computer systems, it is nearly impossible to design a system from scratch and control every details of the system purely by human brain (Simon, 1996). It is difficult to control millions of transactions occurring in a large-scale E-market. It is also difficult to monitor an enterprise information system which encompasses huge amounts of heterogeneous devices which covers thousands of different geographical locations (Rothkopf, 2003) (Coulouris et al., 2000*a*). Grid Computing, Autonomous Computing, Pervasive computing and Multi-agent systems, are all committing themselves to challenging the design of large-scale distributed system (Coulouris et al., 2000*b*). Computational trust is to make an intelligent agent trust another agent and delegate part of their tasks to the target agent in a heterogeneous distributed multi-agent environment. Delegation of action is the result of trust and it

also forms the foundation of future large-scale cooperative computer systems. Generally, trust toward specific agent is generated through recognition and experience under repeated transactions with that agent. Reputation is the socialized trust which can be propagated through a social network of agents. It helps agents trust the target agent without any direct interaction with the target agent. The benefits of introducing trust and reputation into multi-agent system include:

- As a lubricant, trust can eliminate much of unnecessary communications which are currently necessitates many interaction protocols thus greatly improve the performance of the multi-agent systems.
- An agent can make decision easier based upon the evaluation of the trustworthiness of another agent. Computational trust is also a very beneficial addition to the traditional decision theory.

Trust is a kind of soft security which complements the traditional hard security like encryption, authorization, and authentication. An agent that exists in complex heterogeneous environment must possess both securities in order to be safe and effective.

The mechanisms for coordinating interactions among agents are always pre-defined, that is, the designer specifies how one agent responses to another agent in a fixed protocol (Huynh, 2006). Such mechanisms are not flexible enough because of the intrinsic high openness and variability of the distributed systems (J.Ferber, 1999). For example, in open MAS (Multi-agent Systems), an agent cannot expect to always interact with the agents in the predefined application domain in a predetermined way (Subrahmanian et al., 2000). Agents will interact with different agents coming from heterogeneous applications and they may face challenges from lying, deceiving and accidental incidents (J.Ferber, 1999). Such complexity creates the following questions: can agents accept services from other unfamiliar agents? Can agents make use of the interaction history and transform them into experiences? Can agents avoid excessive negotiation with a familiar agent in an efficient way? Computational trust seems to be the answer and the next step of research for the multi-agent systems. Thus, a systematic review on the existing trust models is necessary.

Trust actually is a belief that someone or agents can delegate the host to finish some actions. There are two layers of meaning in the expression: first, agents should

generate the belief of trustworthiness toward some other agents in some specific form; second, agents should make decision whether to delegate actions to the trusted agent. The first layer is actually to study how agents generate and update their belief which is part of research from computational intelligence. The second layer is an extension of the traditional decision theory which adds agents belief and trustworthiness as one of the concerns during decision making.

There are a few computational trust models and reputation models that have been proposed from (Subrahmanian et al., 2000) to (Huynh, 2006). From a point of dimensional view, there are two types of models involved, i.e. local trust based and reputation based models. For local trust based model, the early model developed by Marsh, University of Stirling, 1994, only considers the local trust dimension which only derives trust from the agent's direct interaction without referencing to the recommendations from other witness agents (Marsh, 1994*a*). For reputation based models, like SPORA (Zacharia, 1999), they only consider the reputation (witness trust) dimension without looking at the local experience of the agent itself. Recently RegreT, Referral Network and TRAVOS take both local trust and witness trust into account (Luke Teacy et al., 2005; Sabater, 2003). They combine the value of the two dimensions with relative weights and finally get a sum. Some models, such as FIRE, even introduce additional dimension called as role-based trust and certified reputation (Huynh, 2006; Huynh et al., 2006).

From a point of algorithmic view, different ways of calculating trust and reputation are proposed. For example, Bayesian systems take binary ratings as input and are based on calculating reputation scores by statistical updating of beta probability density functions. Such models include TRAVOS (Luke, 2006; Luke Teacy et al., 2005). Models such as RegreT (Sabater, 2003; Sabater and Sierra, 2001) are based on discrete trust model to represent trustworthiness and untrustworthiness as a discrete value. A detailed analysis about the composing elements of trust and reputation model will be given in Section 3.

The objective of the study is finding out the current situation and future trends of computational trust for multi-agent systems. It also aims at looking for common necessary compositional elements that compose the models through extracting the essence of those representative models and summarizing their common weaknesses through com-

parison, discussion and analysis. Finally, a clear research path is proposed to help the community to promote the research of computational trust.

2.2 Background structure

2.3 Related technologies

2.3.1 Distributed artificial intelligence

Intelligent computing and distributed computing are the future direction of software design. The Distributed Artificial Intelligence(DAI) is a crossing discipline of Artificial intelligence and distributed computing. The Distributed Artificial Intelligence researchers are both from AI area and distributed computing area. An important issue in DAI is the interoperability, that is, the capabilities to exchange information among programs and cooperations of heterogeneous programs in a fast changing environment.

In DAI, the intelligence is not an independently existing concept, it only exists in a group of agents. The interaction and cooperation among agents are the major research areas in DAI. DAI can be divided into two parts: Distributed Problem Solving (DPS) and Multi-agent System (MAS). DPS mainly considers how to decompose a problem into tasks that can be allocated to cooperating and knowledge sharing modules or nodes. The multi-agent system mainly focuses on the coordination of behaviors among autonomous agents.

Both fields need to investigate the division of knowledge, resource and control. However there is big difference between DPS and MAS. In DPS, there are only one conceptual model, one global problem, and success criteria; while in MAS, there are multiple local conceptual models, problems and success criteria. The purpose of DPS research is to create a coarse grained cooperating module. The modules in DPS cooperate to solve a problem. The problem is usually decomposed into several tasks. Each task is executed by a specialized module. All the policies of interactions are integrated as a whole. DPS is a top down approach of designing systems. On the contrary, MAS is a bottom up approach of designing systems. There are no global controls or global criteria for agents. The dispersed agents are defined first and then the problem is decomposed into several tasks which can be assigned to different agents. The relationship among agents may be cooperative, competitive or even hostile.

2.3.2 Intelligent agent

2.3.2.1 Concepts

Although the word "agent" is widely used in different areas, "agent" is still an ambiguous concept. It is difficult to find a definition accepted by all researchers. The most common viewpoint is that an agent is a computer system designed for some purpose and the system is autonomous and flexible. Wooldridge and Jennings summarize the work done in this field by pointing out that it is better to understand agent from the broad sense and the narrow sense (Wooldridge, 2002).

From the broad sense, nearly all agent based software or hardware have the following characteristics:

- Autonomy: agents are not controlled by humans or other programs. They have control over the behaviors and the internal states.
- Social ability: agents can exchange information through agent communication languages. This is also called communicability.
- Reactivity: agents can sense from the environment and also influence the environment. Whether the agents in real world or the agent in virtual society, they need to sense their environments and change their environments. A program which cannot influence the environment cannot be called as an agent.
- Pro-activeness: the traditional application is passively run by the users and strictly follows the instructions from the users. Agents behave actively or spontaneously. Through sensing the changes in the environments, agents can perform some goal directed behaviors.

Some researchers believe that agents should have some human like characteristics such as knowledge, belief, intention, and desire. Some researchers even propose emotional agents (Bates, 1994; Bates et al., 1992). According to (Shoham, 1992, 1993), an agent is an entity with states composed of belief, capability, choice, commitment and mental component.

2.3.2.2 Theoretical models

Mind elements In order to adapt to the environment and cooperate to solve problems, intelligent agents must use knowledge to change its internal status, that is, the mental state. The mental state drives the actions of the agents. It is possible to build an agent mental state from three aspects: emotion, feeling, and intention. The most famous model based on formal method is the BDI (Belief, Desire, and Intention) model. Belief represents the basic opinions owned by agents toward the environment; an agent also can use it to represent the future possible states. Desire is derived from belief. Desire includes the estimation of the future for agents. Intention confines the activities of agents; it is the subset of goals.

Logic is the most important tool to depict the mental state. Moore is one of the first researchers who used formal logic to model agents (Moore, 1990). His research focuses on the relationship between knowledge owned by agents and the realized activities. He introduces modal operator of knowledge into logic and use dynamic logic to represent actions. Such formalization allows agents to own beliefs about how to achieve goals. And agents can acquire information through executing actions. The disadvantage is his modeling is too simple and incomplete. Cohen and Levesque designed a formal model which is based on linear sequential logic (Cohen and Levesque, 1990). The model introduces concepts like time, event, behavior, goal, belief, and intention. The model also depicts the relationship among these concepts. However, the model does not reveal the relationship between intention and goal, and the model is also very abstract and difficult to implement. Rao and Georgeff propose the BDI model of rational agent based on nonlinear branch temporal logic (S. and P., 1991*a*). They use three modal operators, belief, desire and intention, to build a BDI model. Singh uses an agent logic to describe intention, belief, knowledge and communication. Its logic is also based on branch temporal logic (Singh, 1994).

Rational agent The behavior of rational agent cannot be driven directly from belief, desire or their combination. There should be an intention between the desire and the plan. The reason is that the behavior of agents is confined by limited resources. Once an agent decides to do some task, it needs to build a commitment with limited form. In a multi-agent environment, there must be a commitment to coordinate the behaviors of agents. Without commitment, there are no actions. Intention is just one type of

commitments. In an open environment, a rational agent's behavior must be restricted by intention. If an agent changes its own intention, the changes must have some reasons. An agent cannot ignore the changes in environment and keep unimportant or not realistic activities.

2.3.2.3 Structures

An agent can be looked as a black box with sensors and effecters. Agents use sensors to sense the environment and use effecters to change environment. Except when interacting with environment, agents need process and explain information received from outside. The information is merged and processed into understandable data. There are three types of agent structures: deliberative agent, reactive agent, and mixed structure.

Deliberative agent is also called cognitive agent. It is an explicit symbol based model which has abilities of logical deductions (Wooldridge, 2002). It is a knowledge based system that keeps the essence of the traditional artificial intelligence. There is an environment model implemented as the major component of the knowledge library. Agents using this structure will face two basic problems (Castelfranchi, 1999): (1) transformation problem: how to translate the real world into an accurate, appropriate symbol description; (2) Representation/deduction problem: how to use the symbols to represent the entities and processes in the real world, and how to let agents make decision through deducing the information. The BDI model (Kinny et al., 1996) is a typical deliberative agent structure. The biggest disadvantage for deliberative agent is its fixed structure. In a complex, fast changing environment, it is too stationery to be reactive. The reactive agent is an agent without complex symbol deduction and symbol based world model (Ana Lilia Laureano Cruces, 2000). It uses the condition-action rule to connect perception and action.

Neither the symbol base deliberative agent nor condition-action based reactive agent can be built into a perfect agent. Some researchers propose the mixed structure in order to combine the advantages of both types. The most direct method is to build two sub systems in an agent: one for deliberative structure, the other for reactive structure. Usually the reactive sub system has higher priority than the deliberative sub system because the agent needs to react to important event in real-time. A typical mixed structure base agent system is MAPE (Zhongzhi et al., 1994). Each agent contains a series of modules including sensory, action, reaction, modeling, planning,

communication and decision making. The agent has an agent kernel which allows additional modules be attached to the kernel dynamically. The kernel is made up of internal database, mail box, and blackboard and execution engine.

2.3.2.4 Communications

The commonly used communication methods include the black board system and the message/dialog system. The black board system provides a public working area; agents can exchange information, data, and knowledge there. The black board is the medium between agents. Agents do not communicate with each other directly. Message based communication is the foundation of realizing complex coordination policies. Through the specified protocol, agents can exchange messages to build communication and coordination. It is different from the black board system in that the messages are directly exchanged between two agents without any caches. Message is sent to one specific agent or specific group. Only the agent or group with that address can read the message. The protocol must specify the process of communication, format of messages, and languages of communication. Of course, agents must understand the semantics of the language.

Knowledge Query Manipulation Language(KQML) is a uniform agent communication language (Labrou and Finin, 1997). KQML specifies the format of messages and the message transmitting system. It also provides a group of protocols for recognizing, establishing connection and exchanging messages. But the semantics of message content is not specified clearly in KQML. KQML can be analyzed from three layers: communication, message and content. The communication layer specifies the techniques, and the parameters of communications. The message layer specifies message related types of speech acts. The content layer specifies the content.

FIPA (Foundation for Intelligent Physical Agents) defines a language and designs supporting tools. The language is called Agent Communication Language (ACL) (Labrou et al., 1999). The thoughts of ACL are originated from interaction of human society. It does not specify any low level service like network protocol or transmitting services. It assumes the physical infrastructure exists. The characteristics of ACL including: defining goal driven behaviour, defining autonomous decision making processes for agent actions, building interaction protocol through negotiation and

delegation, creating models of mind states, and enhancing the adaptability to environments and requirements. In this thesis, ACL is followed as the communication language among agents.

2.3.3 Multi-agent system

2.3.3.1 Coordination and Cooperation

Coordination and cooperation is the core research topics for multi-agent systems. It is also the major difference between multi-agent systems and the other types of computer systems. The purpose of build system in a multi-agent way is to coordinate the knowledge, desire, intention, planning, and actions so that the agents can cooperate to complete the tasks.

Coordination is to change the intention of the agents. Cooperation is a special case of coordination. In an open dynamic environment, agents with different goals need to coordinate its goals and resources. If there is a conflict in using resources, without a good coordination, there will be a dead lock. In another scenario, if a single agent cannot finish its goal, it may need helps form other agents, this requires coordination.

Cooperation can improve the performance of the multi-agent system and enhance its capability to solve problems. It also improves the flexibility of the system. While individual agents are independent from other agents and only cares about its own needs and goals, when in a multi-agent system, agents must follow the predefined social rules and its behaviors must meet the preset social norms. The interdependency among agents strictly restricts the interaction and cooperation of agents. Based on different mechanisms of interaction and cooperation, the implementation of agents in a multi-agent system is different.

Nowadays, the researches on agent cooperation mainly fall into two types. The first type is to migrate the theories into other domains (Game theory, classical dynamics) for agents (Kraus, 1997). The second type is to research the cooperation from the idea of mind attitude as goals, intension, planning like model named as FA/C (Kraus, 1997), joint intention framework (Levesque, 1990; Wooldridge and Jennings., 1994), and shared planning (Grosz and Kraus, 1996). The first type of mechanisms is not as widely used as the second because once the environment changes, the number of agents, types, and interaction relationships might be different from the situations that

are suitable for the theories. The theory based interaction thus loses its advantages. The second type focuses on problem planning and solving.

2.3.3.2 Contract net

Contract net (Smith, 1980) is widely used in coordination of multi-agent systems. The communications among agents are usually built on agreed message formats. Contract net is actually a contract or a protocol that is based on formats of contract net. The contract net specifies task assignments and roles of agents. It is made up by task processor, contract processor, communication processor and local database.

The local database contains the nodes related knowledge base, information about the current states, and problem solving processes. The other three components use the local database to execute their tasks. The communication processor is responsible for communicating with other nodes. All nodes connect to the network through this processor. The contract processor receives the tasks from bidding. It sends application and finish jobs. It also explains and analyzes the received information. The contract processor executes the coordination among nodes. The task processor is responsible for processing and solving the assigned tasks.

The working process of the contract net is as follows: the problem is decomposed into several sub problems. A node is chosen as the manager to manage the bidding processes. The manager encapsulates the sub problems as bids and publishes them to the nodes. The bids are open to all agents and can be processed through contract processor. The contract processor decides whether to accept the bid. If it accepts, it will inform the manager. The manager chooses the appropriate sub problem for the agent. The agent sends an acknowledge confirming it has accepted the contract. When the processing or solving is completed, the result is sent back to the manager. The essence of the contact net is task distribution. Figure 2.1 illustrates the process of the contract net with a sequence diagram.

2.3.3.3 Cooperation planning

When an agent believes that cooperation bring benefits like improving efficiency or finishing unavailable tasks for single agent, it will have the desire of cooperating and looking for partner to do it. Or when multiple agents find that they can achieve bigger goals, they will form federations and take cooperative actions. The principle of

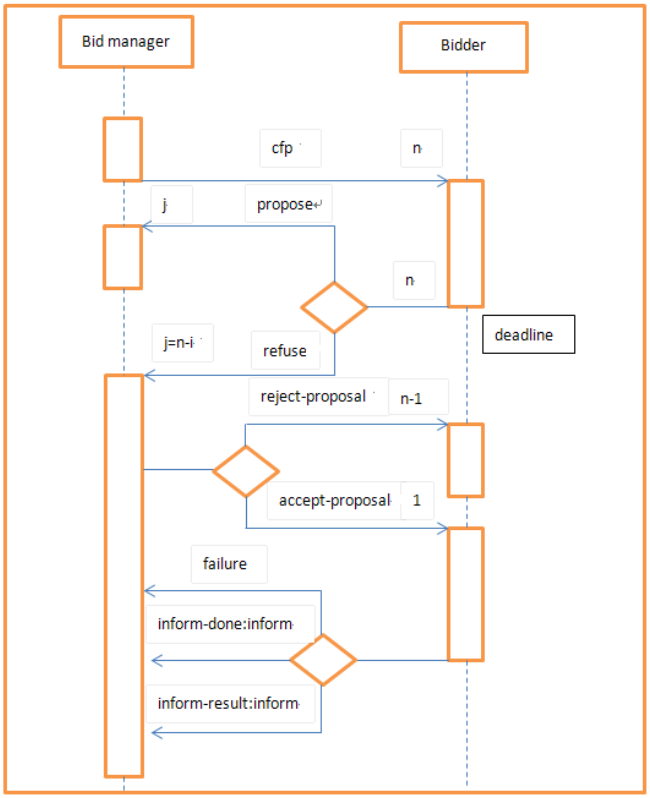

Figure 2.1: - How a contract net works.

cooperation is setting roles on demand and acquiring roles through competition. The roles are preset according to the goals and the needs of cooperation. The agents play specific roles only when they win in the competition attended by multiple agents.

The process of cooperation can be divided into six steps: 1) set goals and produce requirements; 2) cooperation planning and decide cooperation structure; 3) search for cooperating partners; 4) choose a cooperation solution; 5) realize goals; 6) evaluate results.

There are many formal methods to describe the cooperation process. Fox example, d'Inverno uses Z notation (Luck and d'Inverno, 1995), Fisher use sequential logic (Fisher and Wooldride, 1997), Rao uses modal logic base on derived channel (S. and P., 1991*b*). Although the logic based formal method can accurately describe the mind attitude and time series for agents, the action norms depicted by logics is still far away from implementation. The logic method is good at depicting the static properties and social norms, but it is not good at depicting communication and interaction structure among agents. Some researchers use process calculus in process algebra such as π calculus to describe the dynamic behavior processes, concurrent interactions and cooperating structures. Some researcher combines the formal methods and process calculus into an uniform formal structure (Werger, 1999). Using sequential logic to depict goals of cooperation while using π calculus to depict interaction processes and cooperation structures.

2.3.3.4 Ecology based coordination

The distributed computing system has similar social, organic characteristics as living beings. Such open system is different from the current computing system. It can do asynchronous computing against complex tasks; it can generate nodes in different types of machines. Each node may make local decision according incomplete knowledge and delayed information. There is no central control, instead, it uses interaction among nodes to cooperate to solve problems. These characteristics form a concurrent composition. Their interactions, policies and competitions for resources are similar to pure ecology.

Ecology computing looks the computing systems as an ecosystem. It introduces many biological mechanisms such as mutation and evolution. The mutation leads to

genetic variation and create diversities of species with stronger adaptation. The policies of mutation can be a method to improve the ability of agents in artificial intelligence.

Lenat and Brown (Lenat and Brown, 1984) introduce mutation mechanism into their AM and Eurisoko system through grammar mutating in Lisp programs to discover mathematical concepts. They believe the future successful systems should be a series of social systems with abilities of evolution and self organization. Miller and Drexler (K. Eric Drexler, 1988) discuss models of evolution such as biological ecosystem, commodity market and points out the difference between ecosystems and computing systems. They also think that a direct computing market is the most ideal system model. Huberman and Hog (Huberman and Hogg, 1995) propose and analyze the process of the dynamic game. When different processes interact to finish the computing tasks, if there are multiple optional strategies, the dynamic gradual process may turn into nonlinear oscillation and chaos. This means that there are no stable strategies of evolution for computing ecology systems. Rosenschein and Zlotkin (Jeffrey S. Rosenschein, 1994) propose to use static game theory to solve conflicts in multiple computing nodes with different goals (Simon Parsons, 2002).

The most famous ecosystem model includes biological ecosystem model, species evolution model, economic model, and social model of scientific group. The intelligence for the ecosystem usually supersedes the individual intelligence.

2.3.4 Learning

The basic learning mechanism tries to transfer a successful behavior in one circumstance to another similar circumstance. Learning is a process to gain knowledge, accumulate knowledge, improve performance, discover order, and adapt to environments. There are four basic nodes in a learning system: learning node, execution node, knowledge base and environment. The environment provides outside information. The learning node (or learning algorithm) process the information provided by the environment (Liviu Panait, 2005). The knowledge base store the information in some knowledge representation form. The execution node uses the knowledge from the knowledge base to accomplish some tasks and provides the feedback regarding the execution to the learning node. Learning improves the performance of the system.

The quality and level of information that is provided by the environment directly influences the ways and effects of learning. The level of information refers to the degree

of abstracting information. The low level information refers to detailed, concrete and raw information while high level information refers to processed abstract information. The quality of information decides the difficulty of learning. If the information is incorrect, incomplete or imprecise, it is impossible to gain the right knowledge.

The learning node is actually learning methods and algorithms, it is also called the learner. There are many methods and algorithms for machine learning such as inductive learning, learning by analogy, analytic learning, genetic learning and connectionist learning. The knowledge base is place for storing knowledge. It should select the appropriate ways of representing knowledge and it should allow modification and extension. The execution node use knowledge to guide actions and to change environments. The results of the execution are provided to the learning node so that the learning node can be improved. Only through the effect of the environments can the learning algorithm be evaluated.

In agent learning, there are two streams: one is to improve performance of the system through agent learning; the other one is to improve learning efficacy and quality of knowledge through agent learning (Alonso and Noble, 2001).

2.3.4.1 Active learning

When an agent works in an environment, the method of gaining experiences of the autonomous agent is active learning. It is different from passive learning. In passive learning, the teachers or domain experts need to choose groups of training data. In active learning, agents learn the effects of actions or plans and amend their domain knowledge. There two types of learning methods for domain knowledge. The first type is theory amendment system such as EITHER (Ourston and Mooney, 1994), OCCAM (Pazzani and Kibler, 1991), CLIPS-R (Pazzani et al., 1994).These systems are representatives of the passive learning. They have preselected one group of training data which is unrelated to the status in training. The second type is reinforcement learning such as Q-learning, classifiers, back propagation. These types of systems does not explicitly focus on the correctness of the learned domain knowledge. They care about the results of executions. The reinforcement learning is widely used. Agents make a plan based on their current knowledge. The execution of the plan produces results which become a feedback to the learning algorithm.

2.3.4.2 Case based reasoning

In case based reasoning, the situations or problems that the agent face are called target case. The memorized situations or problems are called source case. CBR is a strategy that let agent search source cases according to target cases and uses the source cases to help solve the target cases.

Cases are the fundamentals of knowledge representations in CBR (de Mantaras, 2001). Acquiring cases is easier than acquiring rules. This can greatly simplify the process of knowledge acquisition and reuse the experiences of past processes. Agents do not need to start everything from scratch so that the efficiency of processing new problems is improved. Past successful or failure experiences can provide hints to solving current problems and to avoid future failures.

There are two types of CBR in multi-agent systems: process oriented multi-agent CBR and task oriented multi-agent CBR. The process oriented CBR focuses on the processes. Each process can be implemented as an agent. The process oriented CBR is made up of five agents. The case maintenance agent is responsible for maintaining the case library; The case searching agent is responsible for searching and matching the new case; The case modification agent is responsible for the transformation of the case solutions; The case reuse agent is responsible for dealing with the target case according to the transformed solution; The case storing agent is responsible for putting the new case into the case library.

Task oriented CBR focuses on the tasks instead of processes. It provides a CBR agent with multiple strategies to deal with complex tasks. According to the requirements of the specific tasks, it constructs and organizes agents with different functionalities. Every agent is a small CBR system doing specified tasks. Such agent is called a CBR agent. The functionalities of the agents are relatively independent to each other and these agents have their own unique capabilities. Multiple CBR agents can cooperate to finish complex tasks through the form of contract net.

2.3.5 Research Areas

Design new theoretical models for intelligent agents is one of the interesting research areas. Nowadays, researchers are trying to build better theoretical agent models. The models should be based on the autonomous agents and should find the balance between

social interactivity and individual intelligence. Some researchers propose to build rational agents and multi-agent system made up of the rational agents. For example, the logic rationality based BDI model (Kinny et al., 1996); the behavior rationality based Rational Agent (Wooldridge, 2000); The MAS sociality based semantics of open information systems and economic methodologies using game theory and decision making theory.

Negotiation mechanisms and organizations are also important topics that attract many researchers. Another important stream is to research the MAS frameworks and MAS related methodologies for the opening dynamic environments. Such research covers negotiations, cooperation, task allocation mechanisms, social rules, filtering strategies, behavior normalization and federations. An agent organization is a structured agent group with clear goals. In agent organization researches, the hotspots include the models for agent organization, the formation and evolution of the organization, rules of organizations and patterns of organizations. Sociality is an important aspect to indicate the social abilities of agents in MAS. The modern organization theory based agent sociality can become the foundation of the agent theories. It also stands for the research direction of agent theories and technologies.

Implementation tools and applications are the research areas that focus on how the theoretical models can be applied into the real world. The implementing technologies for Agent and MAS are also hot research topics. The current technologies are based upon the existing social organization theory and modeling theories. Many tools have been developed to support MAS including experimental test beds, integrated systems, blackboard framework and object oriented concurrent programming OBCP (Briot and Gasser, 1998). The most representative work in this area is the Agent Oriented Program (AOP) program designing (Bresciani et al., 2004). There are lots of applications with logically distributed problems or physically distributed problems. Nowadays, the major application domains include: language processing, industry manufacturing, organizational information systems, air traffic controls, concurrent engineering design, distributed sensor systems, transportation planning, monitoring and robots. With the high speed development of the Internet, the integration technology based on mobile agents and multi-agent also become new hot research spots.

Intelligent agent should have the ability of learning from their environment. Trust and reputation are important mechanisms that support coordination and cooperation.

If agents can evaluate the trust or reputation of another agent, the performance and reliabilities of the coordination and cooperation can be greatly enhanced. If the agent can learn the trust and reputation through its past interaction experiences, the agent can then dynamically adjust its belief about other agents in the fast changing environments. Thus the framework, mechanisms and algorithms that the thesis proposes contributes both to the agent coordination and agent learning. The trust learning algorithms give agents learning ability while the reputation mechanisms improves the quality of coordination and cooperation.

2.4 Existing systems

2.4.1 Method employed

The investigation is carried out through viewing and analyzing conference papers, journal papers and technical reports on computational trust from varieties of sources. The criteria of choosing analyzing target are based on whether the model is representative and whether the model reflects the latest trend of the research. To achieve a comparative result, the first thing is to figure out the common basic com-positional elements of different models. Taking the elements as parameters then compares the above models in the form of table; it will also try to find out the significance of the result table. The results will be organized in the form of spider graph to show the reader a clear relationship between the evaluation criteria. The study will also differentiate research paths of different models through quantitative analysis and statistics.

2.4.2 Investigation results

Results are shown from two approaches, i.e. centralized approach and distributed approach. The centralized approach saves all the rating procedure, storage of reputation, query of reputation, searching of comments to the computer server, while the distributed approach finishes all these jobs by agents themselves.

2.4.2.1 Centralized approach

There are two existing models that take the centralized approach. They are eBay (Ebay, 2012) and SPORAS (Zacharia, 1999). eBay has built a feedback reputation system for its customer-to-Customer web-sites. The goal of designing such a system is to transfer

the trust and reputation mechanism in the real life human market to the internet-based e-Market. SPORAS is a reputation model was developed by Zacharia in MIT, 1999. It is an evolved version of the eBay's online reputation models. In this model, only the most recent rating between two users is considered. Another important characteristic is that users with very high reputation values experience much smaller rating changes after each update than users with a low reputation. SPORAS incorporates a measure of the reliability of the users' reputation based on the standard deviation of reputation values.

2.4.2.2 Distributed approaches

The thesis lists five existing models that take the distributed approaches and analyze their characteristics respectively. They are Marsh's Model (Marsh, 1994*a*), RegreT (Sabater and Sierra, 2001), Referral Reputation (Yu and Singh, 2002, 2003), FIRE (Huynh, 2006; Huynh et al., 2006), and TRAVOS (Luke, 2006; Luke Teacy et al., 2005).

The pioneer work on computational trust model was done by Marsh (Marsh, 1994*a*) in 1994. Marsh thought that knowledge, utility, importance, risk, and perceived competence are important aspects related to trust. He defined three types of trust: dispositional trust, general trust and situational trust. The trust management provided by Marsh does not treat the collection of recommendations provided by other agents; he only models direct trust between two agents. The aspect of risk is dealt with explicitly based on costs and benefits of the considered engagement. The decision making is threshold based. Among other parameters, the cooperation threshold depends on the perceived risk and competence of the possible interaction partner. If the situational trust is above the value calculated for the cooperation threshold, cooperation will take place otherwise it will not. Furthermore, the decision making can be extended by the concept of reciprocity, i.e. if one does another one a favor, it is expected to be compensated at some time.

RegreT (Sabater and Sierra, 2001) takes trust as a multi-facet concept and a combination of pieces of information. In RegreT, reputation is a combinatorial product of individual dimension, social dimension, and ontological dimension. The calculation of reputation is same as the calculation in individual dimension. The only difference is that all the reputation under each sub-ontological dimension should be summarized.

The model deals with three dimensions of trust or reputation. The individual dimension is based on self-made experiences of an agent. The trust values are called direct trust or outcome reputation. The social dimension is based on third party information (witness reputation), the social relationships between agents (neighbor-hood reputation), and the social role of the agents(system reputation). The ontological dimension helps to transfer trust information between related contexts. For all trust values, a measurement of reliability is introduced, which depends on the number of past experience and expected experience (intimate level of interaction), and the variability of the ratings.

The underlying computational framework for Referral Reputation (Yu and Singh, 2002, 2003) is based on Dempster-Shafer theory (Dempster and Arthur, 1968). The model has a well-defined trust network to propagate the trust value from the witnesses, the proposed approach does not concern the uncertainty which surely occurs in the interaction and there is no risk management in this model.

The FIRE model (Huynh, 2006; Huynh et al., 2006) believes that most of the trust information source can be categorized into the four main sources: direct experience, witness information, role based rules and third party references. FIRE integrates all four sources of information and is able to provide trust metrics in a wide variety of situations. The reliability value bases on the rating reliability and deviation reliability to counteract the uncertainty due to instability of agents.

TRAVOS (Trust and Reputation model for Agent-based Virtual OrganizationS) (Luke, 2006; Luke Teacy et al., 2005) is based on probability theory and Bayesian system (DeGroot and Schervish, 2002). The prominent feature of TRAVOS is that it takes confidence into account and the trustor makes decisions based upon the confidence level instead of the trust value. The reputation is not simply added to the direct trust value or confidence value, the introduction of reputation depends on the value of confidence level. If the agent can achieve the minimum confidence level through checking belief of its direct interaction, then it does not have to query other agents. But if the agent cannot find enough confidence, then the agent needs to seek more evidence from the witness.

The researcher needs to build a test-bed for implementing TROVOS model. Such test-bed is also useful for future evaluation of the researchers own model and other models. The test-bed can mimic a general market where agents carry on business

by automatically standing in for humans. Each agent seizes some resources and cost specific amount of resources with time elapsing (charging energy). The lifetime of the agent ends when the agent owns nothing (starving to death). The goal of each agent is to maximize their assets (resources) and keep itself alive as long as possible. Transaction is the only way to accumulate resources and avoid dying because the system is designed to make agents achieve more resources in transaction than the resources elapsing with time. There are two attitudes held by the agents: benevolence and malevolence. Under the attitude of benevolence, the agent tends to give full value to the other agent in a transaction; under the attitude of malevolence, the agent tends to give partial value to the other agent in a transaction.

Before any transaction, an agent should find out a target agent and evaluate the trustworthiness of that agent. To find out a target agent, the usual way is to use Contract Net Protocol, the agent broadcast its needs and the other capable agents who are willing to sell goods will response to the request listing their preferred price and quantities. However, in our design, to simplify the scenario and focus on the effect of trust model, the target agent is selected randomly by the simulation engine. The trustworthiness is calculated by the agent through combining the reputation from other agents and the experience accumulated in past transactions with the target agents. The evaluation engine is based on TRAVOS model. Agents must consult other agents through broadcasting request for reputation. If the agent has knowledge (experience) about the target agent then the agent should respond to the request and send their experience to originated agent. The originated agent then combine these experience with its own experience according to the TRAVOS model to decide whether to make transaction with the target agent.

- Dimension is to study what the sources of the trust values are.
- The semantics focus on the meaning behind the trust in case that the outcome from the trust is a composite product.
- The mathematical model employed in calculating trust or reputation is sometime called trust computation engine (Josang et al., 2007).
- Trust network is the study on the topology of organizing witness agents and the host agent.

- Uncertainty refers to the management of risk which monitors the accidental incident and environmental changing, and reliability which ensures that the trusted agent is reliable enough even though it is trustworthy that is based upon the result of mathematical calculation.

Model Name	Dimension	Semantics	Architecture	Trust network	Uncertainty	
					Reliability	Risk
eBay	Reputation	Single	Centralized	N/A	N/A	N/A
Marsh	Local trust	Single	Distributed	N/A	N/A	N/A
SPORAS	Reputation	Single	Centralized	N/A	Exogenous	N/A
Referal System	Local trust reputation	Single	Distributed	Directed graph with depth limit	N/A	N/A
Regret	Local trust Reputation	Ontology-based	Distributed	N/A	Endogenous	N/A
FIRE	Local trust Reputation Role-based trust Certified trust	Single	Distributed	Directed graph with depth limit	Exogenous	N/A
TRAVOS	Local trust reputation	Single	Distributed	N/A	Exogenous	N/A

Table 2.1: Comparison - The difference among different computational trust models.

2.4.2.3 Observations

Different models are compared in table 2.1. Significant observations from the table can be listed as follows.

1. There are multiple facts (cardinality of dimensions) to forge the trust or reputation. 3 models are single dimension and 4 models are multiple dimensions.

2. 6 models presume the semantics behind trust is consistent to all agents except that RegreT adds an ontological dimension to deal with the semantic difference.

3. 5 models choose distributed architecture rather than centralized architecture.

4. 6 mathematical models are based on summation or product of different dimensions with selected weights representing their influences. TRAVOS is an exception which is based on Bayesian probability theory. (The evaluation criterion is the confidence level instead of the trust or reputation value).

5. 5 models, except Referal System and FIRE, don't take trust network and trust transitivity into account.

Their hypothesis is that trust propagates from the target witness to the host agent without any distortion or loss. The findings are: semantics, risks, trust network pose weak points for most models, whereas the dimension, architecture and mathematical models are intensively studied by researchers.

2.5 Discussions

Most models have two basic dimensions: local trust and reputation. But they aren't the only dimensions that can be used to deduce trust. FIRE (Huynh, 2006; Huynh et al., 2006) introduces role-based reputation, which models the trust to the specific role in the society. This is similar to the situation that people always tend to trust some group of people with occupations like professor, doctor or police in real world society. FIRE (Huynh, 2006; Huynh et al., 2006) also has a design called "certified reputation" which is reputation collected by the trustee from its previous interactions. The truster then does not need to contact its acquaintances to know about the trustee thus the design improves the efficiency of communication. Some idea can be borrowed from social science study on trust; cultural trust and mechanism trust (Sztompka, 2000) are trust that may find their places in computational trust. Their potential hypothesis is that their trust is restricted to a predefined topic. For example, in eBay, trust (reputation) to the seller implicitly means the belief held by the buyer that the seller will send the correct product to the correct location within the right time frame (Ebay, 2012). However, this is an ambiguous semantic. Some sellers may believe that the correct product to the correct location with a little delay deserves the buyer's trust. Such gaps of semantics lead to disruption between buyers and sellers. In multi-agent systems, if two agents with different trust semantics meet, do they simply refuse to

trust the other because of incompatible semantics or they need to build up a consensus through negotiation? If the semantics of trust can be adjusted, how often should such adjustment happen and to what extent? It is necessary to create an ontology built upon XML and RDF that allows systems to provide machine-readable semantic annotations for the trust of specific domain. Most models do not explain how the trust network work and what mechanism the trust transitivity is based. How the trust or reputation value is transferred? Does it simply keep the original value from the witnesses? Or does its value attenuate along distance as same as what happens in the transitivity of human reputation? The solutions to these questions are keys to implement a practical system. Theories about social network analysis in social science make sense to build algorithms for searching witnesses in the network of intelligent agents. The transitivity or propagation algorithm can also benefit from studying the social network analysis. Some researchers have already noticed the transitive trust and proposed their ideas (Josang and Pope, 2005).

None of the reviewed models introduce risk management to control the uncertainty due to the environmental influence or accidental incidents, although they do define reliability to counteract the uncertainty due to instability of individual agent. It is necessary to use risk evaluation to evaluate the risk associated with the prospective transaction under specific environmental facts. The risk here mainly means the risk derivate from the environmental influence or accidental incidents. For example, agent A trust a reliable agent B, but B's environment is instable (B's system often breaks down due to an accidental power off). Such type of risk can be called as environmental reliability. If agent A has assessed the risk of B's unstable environment additional to its reliability and trust, it will be more careful when making decision about whether to trust B or not.

An important part is not listed in the comparison table but deserves discussions. Most models stay as a theoretical model without performing a strict experiment in the real system. A few of them do have simple test but are not complete. A complete experiment to assess the trust model for MAS should at least pass the functionality test, the performance test and the security test. The functionality test focuses on whether the model supports heterogeneous agents to effectively cooperate with each other in different scenarios. The performance test focuses on measuring the efficiency of the model in the form of comparison with other models. The security test needs

to measure whether the trust model enhances or does harm to the traditional security like authorisation and authen-tication. The test environment should firstly be a simple but complete multi-agent system. Some tools of building multi-agent system can be applied to the experiment like JADE (Java Agent Development Environment), FIPA-OS, zeous, etc. The details about different tools can be seen in technical report from Gerstner Laboratory (Laboratory, 2005).

JADE (Bellifemine et al., 2007), FIPA-OS (coordinating team, 2012), and ZEUS (Nwana et al., 1999) are the most prominent and prevalent platforms that support MAS development. The similarities among the three platforms are: all of them are based on Java programming language, all of them are open-source project, and all of them claim to strictly follow the FIPA (Foundation for Intelligent Physical Agents) specification. Through comparing these tools, the following observations and conclusions can be made:

- JADE platform is a better choice for MAS development than FIPA-OS and ZEUS in FIPA compliancy, platform maturity, and non-technical concerns.
- Adopting XML as the specification message encoding, representation, and content language can be a good choice to alleviate interaction problems.
- It is necessary to extend the range of applicability of the agent mobility to low-end devices.
- Web Service enhanced agent can freely integrate with any system that supports Web Services; The combination of agent paradigm and the Web Service paradigm may form an intelligent service provision network to complement the traditional static service oriented architectures (SOA).
- It is necessary to introduce agent oriented software engineering process and agent based modeling tools such as AUML (Luke, 2006) into MAS platforms.

The continuity of development efforts is important for the maturity of a successful agent platform. For the MAS research community, it is possible to develop better tools and platforms to support the agent development through three routes. The first route is to update the FIPA specification with continuously absorbing new findings and new mechanisms, to incorporate with the existing widely adopted specifications such as

XML and UML; the second route is to build more powerful application platform through strengthening the administration, monitoring, debugging and logging functionality, to exploit the successful development tools like eclipse, to incorporate Web Service and agent oriented software engineering; the third route is to maintain and enlarge the open source communities and let more researchers take part in and contribute to the development of platforms. A framework is designed for the computational trust and reputation. In the mental space, agents carry out trust learning through observing the results of actions. Such learning leads to the generation, increasing or decreasing of trustworthiness. The learning process is discussed in the next section in details. Next to the mental space is the decision space where agents use trustworthiness derived in the learning process to make delegation decisions. It is combinatory decision making process in that the risk and utility evaluation are also included. The outcomes of specific transaction are constantly observed by the other agents. The observation will become the input of next round trustworthiness learning. Apart from the agents own experience, the trust from the other agents, reputation, is also part of the input of trust learning. The reputation from an organized social network and its propagation mechanism is also a research topic in future work. Most aforementioned trust and reputation models are mathematically based upon simplistic algebraic summation or explicit statistical deduction through counting the success or failure of historical transactions toward the target agents. Surely these methods can produce reasonable numerical results and then translate them into thresholds which help agents make decisions. However, the assumption that the agents always repeat the same transactions with the same target opponents is impossible in an ever changing complex multi-agent environment.

- First, the semantics are various among objects and it does not make sense to hold some invariable elements as formula to calculate the trustworthy or not trustworthy.
- Second, the count of success or failure of transaction is non-sense when the target of discussion is different. It is impossible to draw an equivalence between a successful coke transaction and a successful airplane transaction.
- Third, there is no sure clear border between trustworthiness and distrust-worthiness, that is, trustworthiness or its opposite should be a pattern generated from the repeated transactions instead of a simplistic value.

The above analysis leads the modeling of trust and reputation to the area of computational intelligence. Actually, trustworthiness is one kind of belief. The research on generation of trustworthiness is to model a specialized kind of belief for the computer agent. In the study of computational intelligence, several streams are popular and focused in recent years: neural networks, evolutionary computation, swarm intelligence and fuzzy systems. Since 1970s, neural network becomes one of the main research streams of computational intelligence. It is widely applied in machine learning, pattern recognition, and biological science. The reasons of choosing neural network as the basic calculating mechanism are listed as below:

- Instances are represented by many attribute-value pairs. The target function to be learned is defined over instances that can be described by a vector of predefined features.
- The target function output of the computational trust can be discrete-valued, real-valued or a vector of several real or discrete-valued attributes.
- Neural network learning is good choice for fast evaluation of the learned target function. Usually, the recognition of trustworthiness should be finished several times per second by the agents.
- The ability of humans to understand the learned target function is not important. The weighted learned by neural networks are often difficult for humans to interpret.

The input layer is composed by dynamically organized element. The elements are ex-tracted from the semantic library according to the target object. For example, if the target object is digital camera, then the elements extracted maybe price, quality, guarantee and delivery. The neural network will gradually adjust its weights in a way of unsupervised learning.

2.6 Summary

This chapter reviews the background knowledge of intelligent agent and multi-agent systems. It explains the difference between DPS and DAI, and gives DAI a detailed description. Then the intelligent agent is reviewed from its concept, structure and

communication. The chapter explains the concept of agent and discusses the meaning of agent in broad sense and in narrow sense. The chapter lists the popular theoretical models and compares the difference of these models. The rational agent and Belief, Desire, Intention model is the focus of discussion. The intelligent agents are the fundamentals of multi agents. The chapter investigates the coordination and cooperation mechanisms and analyzes the advantages and disadvantages of different mechanisms. The agent learning topics is also covered. The rational of learning is explained and the common algorithms such as active learning and CBR are mentioned. At the end of the chapter, the researcher lists some interest topics in the field and state how the thesis related research results locate in the background.

Plenty of interests have been attracted to the construction of computational trust from various research communities. Through analyzing and comprising theses models, the thesis proposes that a complete computational trust model should at least have seven fundamental elements. Some conclusions are drawn as follows:

- The current trust dimensions are not enough to represent trust in multi-agent systems and new dimension can be modeled through introducing concepts in sociology.
- Agents from different domains must fill their semantic gaps through constructing ontology with XML and RDF.
- The searching algorithm for trust network and the propagation mechanism for trust network can be progressed through introducing techniques in social network analysis.
- Except for the reliability of target agents, agents also need to manage the risk (or environmental reliability) due to environmental changes or accidental incidents.
- A complete experimental platform which used to test the functionality, performance and security of computational trust model is a necessity.
- The future work of the research domain can also be naturally derived from the above conclusions.

- The dimension of trust can be extended based on the study of trust in sociology and psychology. It is necessary to categorize the scenarios where the dimension is appropriately used.
- The results in the research of semantic web can be used in the computational trust model. It is necessary to create an ontology built upon XML and RDF that allows systems to provide machine-readable semantic annotations for trust of specific domains.
- Theories about social analysis in social science can be used to build algorithms for searching witnesses in the network of intelligent agents. It is necessary to develop a trust transitivity or propagation mechanism which fulfills the requirement of different situations.
- There is still a need for a simplified, effective mathematical model which can be evaluated through appropriately constructing experiments.

3

Neural Trust Model

3.1 Introduction

The problems found in the existing models push the researcher to look for a better solution for computational trust and computational reputation. According the problem exposed in the previous chapter, the newly proposed model should be a systematic model which supports both trust and reputation. The model should also take the learning capability for agents into consideration because agents cannot quickly adapt to the changes without learning. The model also needs to have the ability to make decisions according to its recognition of trust.

Before actually building the model, it is necessary to analyze the concept of trust. Usually when people say trust they mean human trust, however, in this thesis trust refers to computational trust. How human trust is different from computational trust is a very interesting question. The answers to the question helped the researcher recover many features of computational trust and built a solid theoretical foundation for the proposed model. This chapter tries to compare the definitions of trust in different disciplines such as economy, sociology and psychology. This chapter also tries to make a definition of computational trust and analyzes such trust from several different perspectives.

The description of the model is important. As a whole, it is represented as a framework that defines components and component relationships. As the concrete components, the purposes and responsibilities of the specific component are explained. This is to illustrate the static structure of the model. The dynamic structure of the

model is described as the process of executing the model.

There are three main parts in this chapter: section 2, section 3 and section 4. Section 2 analyzes the concepts of trust from different aspects and figures out the core features. Section 3 discusses the considerations of designing a computational trust model. Section 4 presents the framework design of the model, and it illustrates the components and describes the relationship of the components. Section 5 summarizes this chapter.

3.2 Concept analysis

3.2.1 Trust in human society

In any dictionary, trust is defined both as a noun and as a verb. In this section, three major dictionaries are chosen to gain insight into the meaning. In Oxford online dictionary (Oxford, 2011), the word trust is defined as: "1) (noun) firm belief in the reliability, truth, or ability of someone or something; 2) (noun) an arrangement whereby a person (a trustee) holds property as its nominal owner for the good of one or more beneficiaries; 3) (verb) believe in the reliability, truth, or ability of "; In the Merriam-Webster online dictionary (Merriam-Webster, 2011), the word trust is defined as: "1) assured reliance on the character, ability, strength, or truth of someone or something; 2) dependence on something future or contingent; 3) a property interest held by one person for the benefit of another; 4) a charge or duty imposed in faith or confidence or as a condition of some relationship. 5) (verb) to place confidence ". In the Longman online dictionary (Longman, 2011), the word trust is defined as: "1) a strong belief in the honesty, goodness etc of someone or something; 2) (verb) to believe that someone is honest or will not do anything bad or wrong; 3) facts/judgment to be sure that something is correct or right ".

All these definitions have something in common: First, in essence, trust is a human belief which is held by the host. Second, the object or the target of the trust is the trustee. The contents of the trust is the ability of trustee, and the truth told by the trustee or the reliability of the trustee. Third, the relationship between the host and the trustee is reliance or dependence. The host delegates its benefit related activities to the trustee and relies on or depends on the trustee.

The typical definition of trust in sociology (Mayer et al., 1995) follows the general intuition about trust and contains such elements as: the willingness of one party (trustor) to rely on the actions of another party (trustee); reasonable expectation (confidence) of the trustor that the trustee will behave in a way beneficial to the trustor; risk of harm to the trustor if the trustee will not behave accordingly; and the absence of trustor's enforcement or control over actions performed by the trustee (**?**).

In psychology, according to the psychoanalyst Erik Erikson, trust is believing that the person whom is trusted will do what is expected (Cofta, 2007). Trust is integral to the idea of social influence: it is easier to influence or persuade someone who is trusting. The notion of trust is increasingly adopted to predict acceptance of behaviors by others, institutions (e.g. government agencies) and objects such as machines. However, once again, the perception of honesty, competence and value similarity (slightly similar to benevolence) are essential.

In Economics, trust is also seen as an economic lubricant, reducing the cost of transactions, enabling new forms of cooperation and generally furthering business activities, employment and prosperity. This observation created a significant interest in considering trust as a form of social capital and has led research into closer understanding of the process of creation and distribution of such capital (Fukuyama, 1996).

All the above discussion about trust refers to human trust or trust in human society. For a multi-agent system, thousands of autonomous agents interact with each other without a moment's pause and also create or emerge as a virtual society. These agents are proactive agents with clear goal to pursue. As an important social mechanism, trust can also play a critical role in the agent society to promote the effectiveness of the interactions. Such trust for an agent or for a computer program can be called "computational trust". Computational trust is a simulation of human trust but with its own unique characteristics. In the next section, the concept of the computational trust will be explained in details.

3.2.2 Computational trust

3.2.2.1 Definition

After analyzing the concept of trust from different aspects and disciplines, the thesis needs to make its own definition so that this definition can be the foundation of further

discussions. The definition of the computational trust in this thesis is defined as:

Computational trust is a belief that an agent (trustor) perceived for another agent (trustee) in a specific domain and in a specific duration, the belief is that the trustee is honest, reliable and capable of delegating the trustor's benefit. Namely, the information provided by the trustee is true; the trustee has the ability to successfully finish the trustor's delegated jobs; the trustee is a reliable partner to cooperate in specific tasks.

There are some important composing words for this definition: belief, trustor, trustee, object, duration. These words also reflect some features of the computational trust. Figure 3.1 illustrates the features of the computational trust.

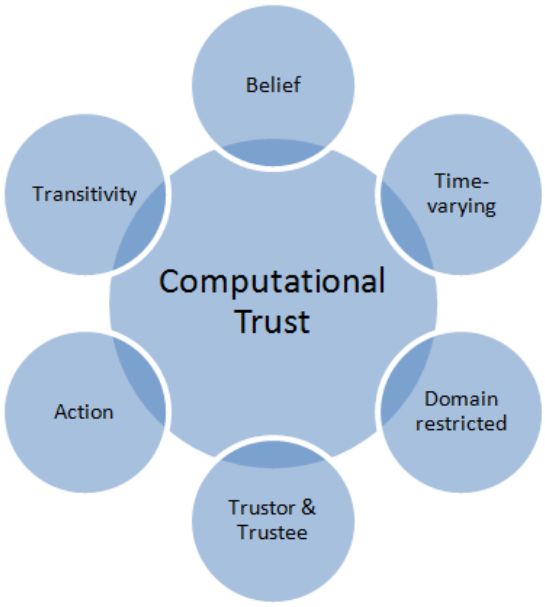

Figure 3.1: Features - The features of the computational trust

- Belief:Belief is introduced into many agent design models. It is also the core part of Belief, Desire and Intention (BDI) model (Kinny et al., 1996). The computational trust is a special type of belief which only focuses on social agents in a virtual society.

- Trustor and trustee: Every trust relationship must have a pair of trustor and trustee. Trustor and trustee are just role name in a trust relationship. In a

virtual society, an agent could be a trustor in one scenario and be a trustee in another scenario. Such role allocation also implies a direction in the trust relationship. If a line with arrow is drawed to represent the relationship, then the arrow should point to the trustee form the trustor. Take an electronic transaction as an example. The buyer agent is the trustor who will trust the seller agent. The seller agent is the trustee who will be trusted by the buyer agent.

- Domain restricted: The computational trust is domain restricted because the trust without domain is nonsense and impossible to operate. An agent trusts another agent only on some specific domain. A domain is just a concept of set or collection. It reveals a fact that an agent is trusted only in its familiar scope. A domain can be an industry, a type of product, a geographical area, etc. It is possible to generalise the model in the future through introducing an ontology layer which supports the semantics from multiple domains so that the agent can carry out trust recognition tasks in different standard domains. Take figure 3.2 as an example. In a electronic market, Agent A trust Agent B in the domain of publishing and book sales, while Agent A does not trust Agent B in the domain of food and restaurant because B is only a dealer agent for publishing industry.

- Time sensitivity: The computational trust is dynamical, that is, it will not stay in an unchangeable state, it is always changing with the outside environment. Time is a dimension that measures the sequence of state changing. Thus the computational trust can be said as Time sensitivity. The state of the trust is changing as time elapses.

- Transitivity: In a multi-agent context, computational trust is not just a belief in the individual agent. It is a social mechanism. It is generated from the individual agent and propagated to the other agent in the society and so the computational trust is transitive. Most of the time, the transitive message is also called reputation. Of course, it cannot be propagated without any boundary and any loss. The mechanism of reputation is discussed in Chapter 6.

- Action: As a belief, the computational trust will not directly lead to actions between agents. The belief is only knowledge about some specific agent. Such knowledge is only used to support the decision making on whether to start an

action. The computational trust helps an agent decide whether to take actions or give up actions based on the result of trustworthiness or untrustworthiness. Trustworthiness is also an important topic that will be discussed in the next few sections.

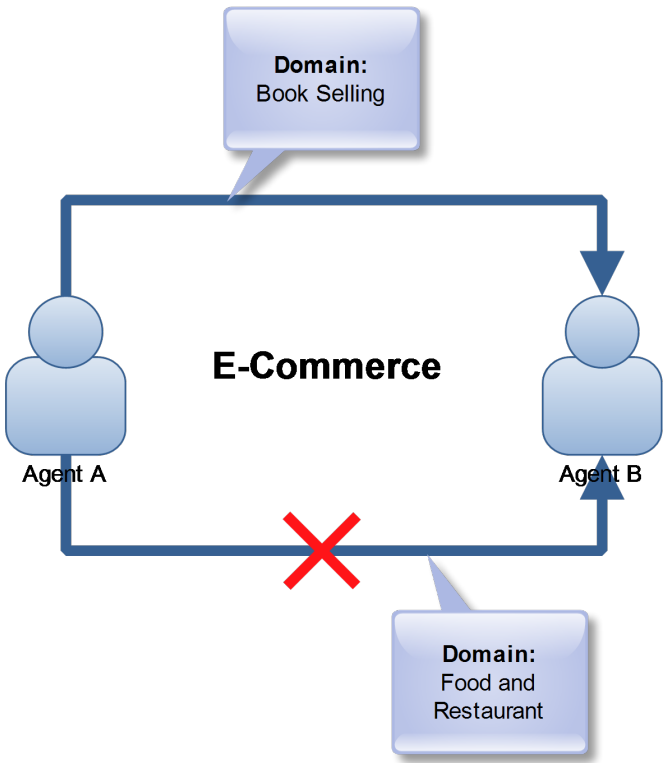

Figure 3.2: Domain difference - Computational trust should be designed to use in a specific domain because the trustworthiness is estimated from different perspective in different domain.

Computational trust can be used in the transactions of Electronic Commerce which are participated by autonomous agents. Agents can choose to trust or not trust its potential trading partner. Such trust may lead to subsequent transactions or denial of transactions depending on the belief of trust. Computational trust can also be used in a cooperation scenario which requires the participating agents to interact with each other. For example, in robot soccer game, whether two agents cooperate with each other to make a shoot or pass depends on whether they have belief of trust in the other one. If they do not have the trust then such cooperation is not considered and they will look for

another agent that they trust to cooperate. In fact, like human societies, computational trust is also an important lubricant that makes the virtual society operate in a more efficient and effective way. Once multiple agents are assembling, there must be a space for the computational agent. Just like a machine, it needs to be lubricated by the machine oil in order to keep its different components running without wearing and tearing.

3.2.3 Computational trust vs. Human trust

The idea of computational trust is borrowed from human societies. Just like the autonomous agents are simulations of human individuals. The computational trust is also a type of simulation based upon human trust. However, the physical foundation of the computational trust is the electronic computer. The living context of agents is an Internet based virtual society which is different from the real world of human being. Such facts decide that computational trust must have some characteristics that human trust does not have. The difference between the two types of trust is deserved to be investigated. Through studying their differences, the researcher can learn the advantages and disadvantages of human trust and what mechanisms those advantages are based upon. The researcher can also see how computational trust avoids the disadvantages of human trust due to its physical limitations. The researcher can obtain the ideas of how to design good mechanisms appropriate for multi-agent systems.

Table 3.1 tries to analyze the differences between the computational trust and the human trust from seven facets. These facets cover the actor involved, the physical and social foundation, the belief model, the capability of storage and transitivity, the range of domains and the applicability.

The involved actors in computational trust are the autonomous agents in multi-agent systems while the involved actors in human trust are the human individuals in the human society. Both forms of trust can use the same roles to represent the functions played by the actors, that is, the trustor who generates the trust and the trustee who is endowed with the trust.

All autonomous agents are actually computer programs in the form of agents. The physical representation of computational trust is electronic computers. These agents are theoretically imperishable although they might have their usage life cycle. Human trust is the emergent product of the human brain, and the human brain is materially

	Computational Trust	**Human Trust**
Involved actors	Autonomous agents (as role of trustors and trustees)	Human individuals (as role of trustors and trustees)
Physical foundation	Programs designed for agents running on electronic computer	Human brain and human body
Social foundation	Internet based virtual society which is highly specialized and motivated.	Individual human and their society
Belief models	Usually based on the simple arithmetic models, the statistic models, or the machine learning models	Human nervous system
Storage capability	Based on the computer memories which can be extended, thus the capability is theoretically unlimited.	The human brain has its biological lifecycle. The memory can be forgotten due to aging. Thus the capability is limited and volatile.
Transitive capability	The transitive speed is high enough to ignore the geographical difference. The information can be intact.	The transitive speed is slow and the information tends to be distorted to form "rumors".
Range of domains	Highly specialized to a specific object, a set of tasks, an industry or a designed virtual society.	Highly general, cover all the areas of the human cognition. The trust in human tends to be more generalized and can be applied across different domains.

Table 3.1: Computational Trust vs. Human Trust - The difference between computational trust and human trust.

supported by the human body. Such brain and body are mortal and suffer from aging and death.

The Internet is the basis of multi-agent systems. It connects agents from different applications, autonomous domains or virtual societies. It is the communication foundation and the protocol foundation of the heterogeneous agents. For human trust, the real society connects every person. Human individuals use languages to communicate with each other. The trust emerged from the brain is also expressed and propagated through messages in the form of languages.

Whether in dictionary explanations for human trust or in the above explanations for the computational trust, they both define the trust as a belief. The only difference is one is a belief emerged from the biological brain, the other is a belief represented by electronic circuits. This now also brings the researcher to the most difficult question: what is a belief and how to represent it? Until now, the mystery of human brain has not been fully explored by the neurological science (Ralph, 1999). For computational trust, the simplest way to represent the belief is to use an arithmetic equation to reflect it. The statistic method is also widely used to express the belief in a statistical way. Machine learning is another way which emphasizes the computational trust as a belief that can be learned. The human trust is emerged from the human nervous system. As the other type of beliefs, the trust belief is the product of connection of millions of neurons (Adolphs, 2002).

There is a big difference between the computational trust and the human trust in the facet of the storage capabilities. The computational trust can be stored in the computer's secondary storage as binary data while the human trust can only be stored in some part of the brain. The computer is theoretically everlasting and the memory is theoretically infinite. This means computer based agents can memorize every single detail that is related to the computational trust. Human memory is much less powerful and often suffers from the forgotten due to aging and death.

The transitivity capability is also the strength of the computational trust. The message about trust (reputation) propagated along the network. The speed of transmission equals to the speed of electrons without considering the time spent for the congestion and routing. The accuracy of the message is also pretty good because there is no loss or intended distortion even after it passes through thousands of nodes. The human trust is

unavoidably limited by its physical foundation. The speed of human to human transmission is slow even after using the modern communication technologies. The more serious problem is inaccuracy: the message (reputation) sent out is always magnified, dwindled, or distorted because of intended or unintended reasons. The spreading of rumors is a typical example of the problematic human reputation propagation.

There is also big gap between these two types of trust in the range of domain. For computational trust, it is highly specialized and motivated for some specific domain. For example, an agent designed to be used in a game will not have the same domain of trust as an agent designed for an e-business scenario. However, the human trust is different, it is much more general. Sometimes such generality may lead to some interesting trust transferring phenomenon. For example, If a person trusts a movie actor or actress because of his or her performance, they will transfer such trust to the product that the actor or actress is endorsing. In this example, the domain of trust has been shifted from the movie into the commercial product. The reason of the gap is that the computational trust is a designed trust whereas the human trust is a natural product which is emerged from the human society. Trust transferring phenomenon are useful for agents in some scenario. For example, when agents learn trust from a domain of the desktop computer transaction, such trust is valuable referential experience for a domain of the computer peripherals in case two domains belong to semantically related ontologies.

In a word, there is the necessity to simulate the human trust in the multi-agent system and virtual society in order to lubricate the interaction of agents. The computational trust can apply many ideas and mechanisms which appear in human society but that does not mean computational trust must strictly follow the ways of human trust. Strictly following human's way only leads to simulated human trust instead of computational trust. In fact, the transitivity and storage capability can be designed as even more powerful than the human trust. The best solution is to select the appropriate idea from human trust while rejecting inappropriate. Thus the researcher makes a list which states the points that the computational trust can learn from the human trust and another list which states the points that the computational trust can improve based on the human trust:

- Points can be borrowed:

  - Belief including trust belief is generated from human nervous system. The mechanism can be explained from theory of neuroscience. This reminds the researcher that the computational trust can also be designed in technology of artificial neural network.

- Points need improvement:

  - The ability of speedy transitivity for the computational trust should be promoted. The reputation network can be extended to a reasonably far distance. The computational trust should be propagated without any loss and distortion. There should not be things like "rumors" in the virtual society.

  - The autonomous agent should remember all the information that is related to its trust belief because it does have such storage space and extending potentials. There is no such thing as "forgetting" for agents. The computational trust is based upon all the interaction experiences that the agent might have in its life cycle.

  - For each domain, an agent needs one type of trust to support. Agents can freely extend its domains in case they need to take part in additional domains. The human trust can be transitive between domains. Trust to a good performance of a movie star can be transferred to trust to the product represented by the movie star. However it is not necessary for agent to behave like human in this transitivity issue. Agent can have ontology designed for standardizing the trust domain and the related semantics.

3.2.4 Trustworthiness and untrustworthiness

In Oxford online dictionary (Oxford, 2011), the word "trustworthy" means: "able to be relied on as honest or truthful". In this thesis, the trustworthiness and the untrustworthiness are defined as: they are decisions that are made by the autonomous agent based on the belief of computational trust. The positive decision, i.e., trustworthiness, will lead to actions of agent cooperation, delegation or transaction, while the negative decision, i.e., untrustworthiness will stop any further actions.

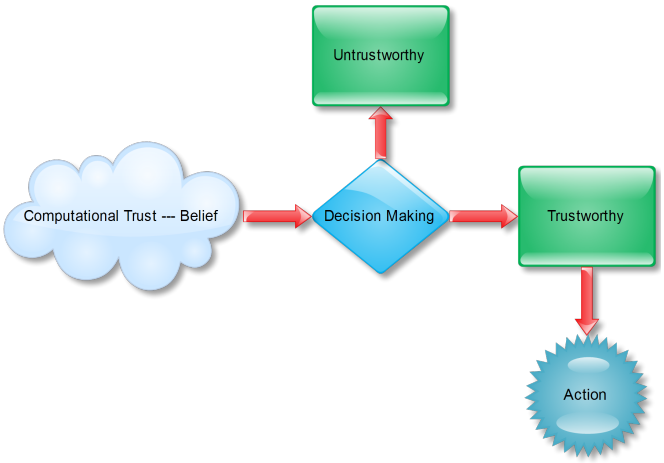

Figure 3.3: Relationship - The relationship between computational trust and trustworthiness.

Figure 3.3 illustrates the relationship between the computational trust and the trustworthiness/untrustworthiness. The computational trust provides a belief basis for decision making. When an agent needs to start a task that involves another agent, the host agent first initiates a trustworthy judgment. It checks its belief to see whether the specific target agent can be trusted. The result is either trustworthiness that allows the agent to carry out its action or untrustworthiness that forbids the agent to go ahead.

3.2.5 Reputation

In Oxford online dictionary (Oxford, 2011), the word "reputation" means: "the beliefs or opinions that are generally held about someone or something, or a widespread belief that someone or something has a particular characteristic". In this thesis, the computational reputation is defined as: a message that is transmitted between agents. The content of the message is information about the trustworthiness or untrustworthiness of another specific target agent. Such information is generated according to the belief of trust hold by the initiator agent toward the target agent. The purpose of the reputation is to give recommendation to the other agents who lack the direct interaction experience with the target agent so that the target agent can make better decision with ease. Thus the reputation can increase the efficiency of the whole system.

The computational reputation is the byproduct of the computational trust. But it does not equal to the computational trust. Because the trust is an individual belief that

needs to be generated from many times of direct interactions, the reputation is only a recommendation which cannot substitute the agents' own experiences and cognitions in their special environments. The reputation can influence the formation of the belief of trust and vice versa. It can be looked as an influential element of the trust belief, though such element is introduced as indirect experiences from other agents.

The transitivity (propagation) is the basic characteristic of the computational reputation. The propagation can take many forms. The initiator agent can only send the reputation to its close neighbor or it can send the reputation to all the agents in the system or in the virtual society. The distance of transmission is a design variable for the propagation mechanism. Additionally, whether the reputation can be propagated intact is another consideration. It is reasonable to introduce the concept of decaying to make the reputation transmit more naturally. When the reputation is propagated relatively long, it can decay according with the distance so that the reputation will not be transmitted unrestrictedly. The distance of propagation is controlled by the rate of decaying and its initial strength.

Figure 3.4 is an example of the reputation propagation. Agent A has experiences of transactions, cooperations or delegations with Agent B. In the virtual society, Agent C is preparing to have some type of interactions with Agent B. It needs indirect experience with Agent B to help it enhance successful possibility of the potential interactions. It turns to Agent A for recommendations. According to the belief of trust toward Agent B, Agent A sends out a reputation to Agent C. Agent C receives the reputation and takes it as an influential element to update its own belief of trust. Then according to its updated belief of trust, Agent C decides whether to take further action with Agent B. Furthermore, Agent C also sends the reputation to Agent D when it learns that Agent D might also have the needs to interact with Agent B. But the reputation sent by Agent D is possibly decayed because it is already second layer node relative from Agent A.

3.2.6 Design consideration

3.2.6.1 Learning trust patterns

In multi-agent systems, "Beliefs represent the informational state of the agent, in other words its beliefs about the world (including itself and other agents). Beliefs can also

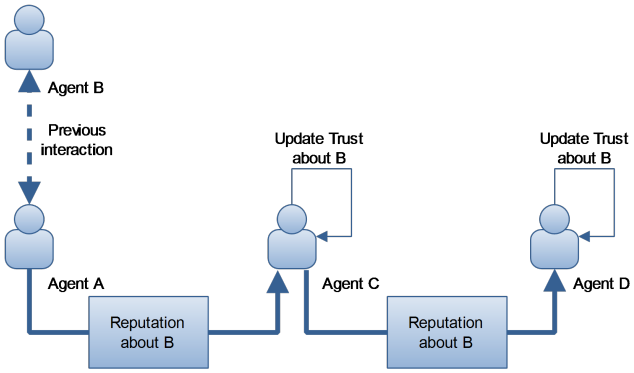

Figure 3.4: Example - The example of reputation propagation

include inference rules, allowing forward chaining to lead to new beliefs. Using the term belief rather than knowledge recognizes that what an agent believes may not necessarily be true (and in fact may change in the future)" (Wikipedia, 2012). From this dentition, it is clear that the belief is also information. Representing beliefs is actually a task of representing the information state of the agent. There are a variety of methods to represent information. In the hardware level, the computers use the binary or octal number system to record information. In the application level, the programs make use of the relational algebra based relational database or make use of the Unicode text file to record information. For information in the belief, these above methods are all applicable. But a more abstract higher level of description is still needed.

In traditional methods like Marsh model (Marsh, 1994*b*), basically, they deem the computational trust as a numerical value. So that they can use mathematical equations or statistical equations to produce a numerical trust value. Such value is also a good indicator for trustworthy level. The decision making is simply to set a threshold for the value: over the threshold, trustworthy; and under the threshold, untrustworthy. For example, in Marsh model (Marsh, 1994*b*), the trust value is calculated using the following equations (Marsh, 1994*b*):

$$T_x(y,\alpha) = U_x(\alpha) \times I_x(\alpha) \times \widehat{T_x(y)}$$

$$CT_x(y,\alpha) = \frac{PerceivedRisk_x(\alpha)}{PerceivedCompetence_x(\alpha,y)} \times I_x(\alpha)$$

In the above equations, the conceived trust value ($CT_x(y,\alpha)$) is the threshold. If the value of $T_x(y,\alpha)$ is greater than or equals to the conceived trust value ($CT_x(y,\alpha)$), an

agent x will cooperate with another agent y. The problem of such traditional methods are: the numerical value is not adaptive enough to reflect the changes of the trust relationship. They are single dimensional instead of multi-dimensional.

To perceive the beliefs of computational trust, agents need to learn from their historical interactions and extract beliefs from the experiences. For computational agents, learning refers to machine learning and the intuitive results of learning are patterns that reflect the emerged characteristic of the interactions. Such characteristics are the informational states that the computational trust requires. And so the thesis proposes to use matrix like pattern learned from agents' interactions to realize the belief of the computational trust. The topological structure of the matrix like pattern can represent the changes of the trust relationship quickly and efficiently. The following matrix is an example which can represent a trust belief.

$$\begin{bmatrix} 0 & -3 & 0 \\ -1 & 8 & 6 \\ -2 & 0 & 1 \end{bmatrix}$$

The next question is which type of machine learning mechanism is appropriate for learning the computational trust and what kind of patterns is suitable for representing the computational trust. The mechanisms used should meet the following requirements of the belief information, especially the trust belief information:

- The patterns of agents' beliefs should be a form that meets the need of frequently mutable and updatable informational state. Agents keep on interacting with each other and every interaction will lead to changes to the informational state of the beliefs. Thus the representing form should record the changes and update the original belief as soon as possible. The best way is to enforce the real-time updating of the agents' beliefs.

- The patterns of agents' beliefs should be a form that meets the need of continuous accumulation of the informational state. Beliefs are deposits of experiences based on thousands of interactions. Every piece of results needs to be reflected on the patterns. The pattern should be compact enough in order to save storage space while it should be complete without losing any details of interactions as well.

- To record the information for each interaction only finishes the first stage of the patterns. The ultimate goal of choosing an appropriate pattern is to find out the

characteristics which exhibit from the form. Because only through finding the characteristics in the belief can agents carry out trust related decision making and actions. Thus the form needs the ability to show the characteristics from the seemingly unrelated data in an organized or emerged way.

- The mechanism that generates the pattern should allow multiple input dimensions because most agents' interactions generate multi-dimensional raw information. The mechanism should transform that raw information to some sensible characteristics.

There are many types of machine learning algorithms. The requirements discussed above support the choice of a neural network as the basic trust learning architecture. The neural network is highly adaptable to the changes in the environments. It can adjust the weight to reflect the changes. This makes the neural network a robust solution for learning computational trust. The pattern generated from neural network can represent the belief precisely. Figure 3.5 illustrates the relationship among trust belief, trust pattern and learning methods.

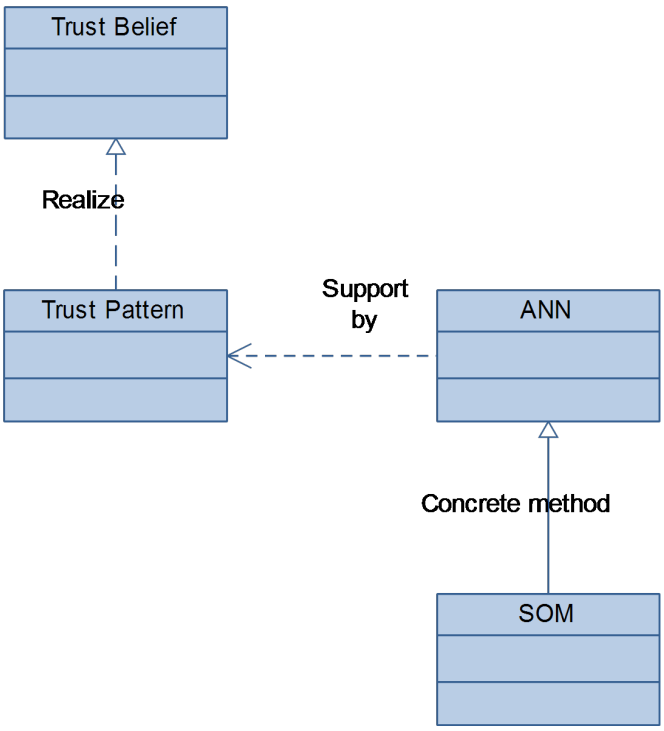

Figure 3.5: Relationship - Trust belief, trust pattern and learning methods

1. Supervised or unsupervised: Most of the time, neural network is used as a supervised learning mechanism. The network is fed with prepared training data and learns typical patterns from these typical data. This is good solution for problems like handwriting recognition and natural language recognition because the typical training data is easy to be organized. However, in the virtual society of autonomous agents, to prepare the typical training data is a much more difficult job to do. The behaviors of agents are versatile. It is difficult to define a standard behavior that is "good" for agents. There are too many types of domains with too many types of agents. There are no consensuses on the definition of "good". Furthermore, the dimensions of the input data for the network changes with different domains. This means that if a set of data in one domain is selected as the training data. Such training data is not representative enough to cover varieties of domains. Thus, the learning of computational trust should be unsupervised learning, that is, there is no prepared training data. The process of learning is totally based on the input data generated from the real time interactions. Each round of the interaction is accompanied with a round of the trust learning. The learning process is all throughout an agent's life cycle.

2. Organized or emerged The representations of the beliefs can be as simple as expressions like "x=x+1" or "x=x-1". The variable x represents the belief of the computational trust for a specific agent. And after each round of the interaction, the host agent accumulates or decreases the variable according to the result of the interaction. A positive result leads to "+1" while the negative result leads to "-1". Such kinds of equations are simple but effective. They become the basic foundations of the trust model for many current successful commercial web sites. Some models also use the statistical theories to construct their computational trusts. In this case, the computational trust is deemed as a mathematical probability. Chapter 3 has already discussed these models in details. Whether the variables or the probabilities are an organized way of representing the computational trust. The mathematical models are applied to abstract the process of the belief generation. In a specific case like Electronic commerce, it works perfectly. However, the mathematical models are not general enough. The researcher has to build a specialized one for each domain or each application. Take the human

brain as a comparison, the belief of the trust is more or less a feeling instead of a real number of rank or a probability. It is an emerged product of the millions of connected neurons in the brain. The autonomous agents need the generality and adaptability like the human brain and the autonomous agents needs to learn trust in their life cycle gradually. The problem of acquiring computational trust should be looked as a problem of machine learning. The computational trust is a pattern that emerged from the experiences of the interactions. There is no fixed formulation to produce trust as in the mathematical models. In this thesis, computational trust tends to be created as a "feeling" of the autonomous agent.

3.2.7 Decision making for trustworthiness

The computational belief are only patterns that reflect some characteristics of the interactions. Actions are not the direct result of the belief. The beliefs are mind level foundations for the action. It is not compulsory that the belief of trust must leads to actions. Only when the autonomous agents have an intention to start an interaction with another specific agent, then the belief is used to support the decision on whether to perform the action or not. The results of the decision as mentioned in previous sections are either trustworthiness or untrustworthiness. The way of using the beliefs of the computational trusts is the interest of the thesis.

In fact, the judgments on trustworthiness or untrustworthiness can be deemed as a problem of decision making. Decision making can be regarded as the mental processes (cognitive process) resulting in the selection of a course of action among several alternative scenarios. Every decision making process produces a final choice (McKnight and Chervany, 1996). The computational trust provides the information that the decision making needs. The result of decision making is the trustworthiness and untrustworthiness that leads to action or rejection.

As mentioned in previous sections, all beliefs of the computational trust are patterns that emerged from the historical interactions between agents. In the human brain, a recent neuro-imaging study (Castelfranchi and Falcone, 2000) found distinctive patterns of neural activation in these regions depending on whether decisions were made on the basis of personal volition or following directions from someone else. Patients with damage to the ventromedial prefrontal cortex have difficulty making advantageous decisions (Mollering, 2005). This research has given the researcher some inspirations

that the decision making for trustworthiness or not can also be based on the distinctive patterns that resides in the memory of the agents. Every interaction occurs will generate a pattern and each of the patterns shows some characteristics of the interactions. If a hypothesis, namely, all the agents are played in the domain, exists, the pattern generated from a successful interaction must be different from the pattern generated from a failed interaction. This difference can exhibit the good features and bad features of the patterns in the interactions. So to make a decision on trustworthiness, an agent can turn to its own beliefs and compare its patterns from positive interactions and from negative interactions. The result of comparison not only tell us whether the target agent is trustworthy or not, but also tell us how much the host agent can trust the target agent because the magnitude of the difference is an indicator how the bad pattern deviates from the good pattern. Figure 3.6 illustrates idea of comparing patterns from successful and failed interactions.

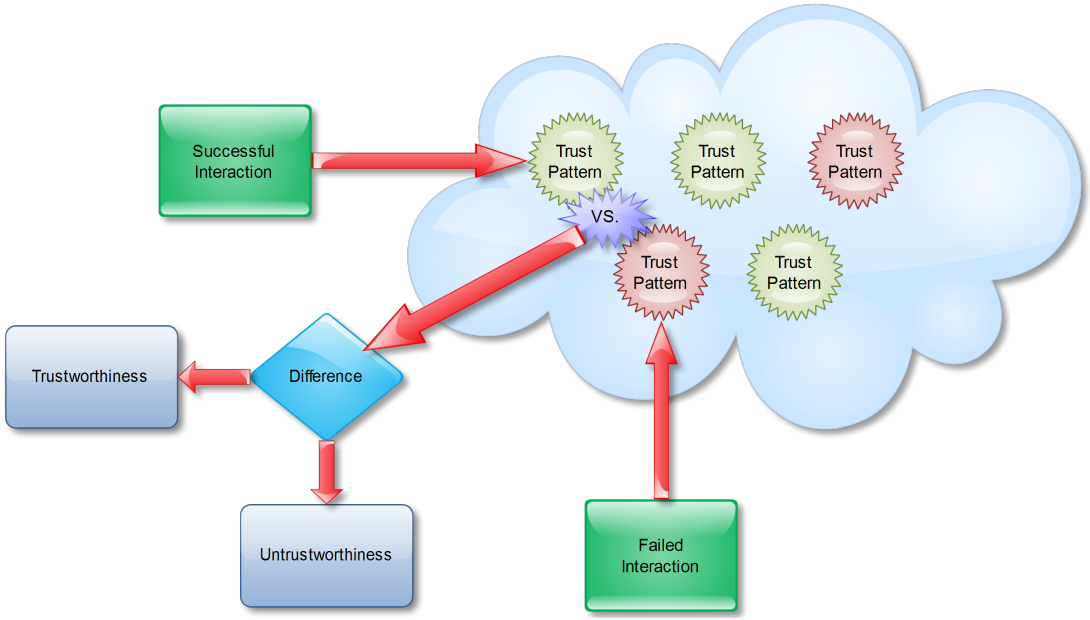

Figure 3.6: Decision making - How comparing patterns help make decision for trustworthy or untrstworthy.

3.2.8 Trust with reputation

The computational reputation is a not synonym of the computational trust. The reputation is a by-product of the society while the trust is an individual belief which is only

suitable for an individual agent. The computational trusts are patterns learned from their interactions. The computational reputations are messages that are transmitted from one agent to another. It is the lubricating oil for the multi-agent based virtual society. The computational reputation is originated from the computational trust. An agent learns trust first and then produces reputation for its partner agents.

The reputation should reflect the host agent's impression toward the target agent. When an agent is asked to give reputation for a specific agent, it needs to check whether it has interacting experiences with the target agent before. If it does not, it stays neutral and rejects sending the reputations. If it has, it might have some patterns that are generated during the interactions with the target agent. To only send out these patterns as the reputation is not complete, because every pattern is a product of a concrete environment with many specific factors and every pattern is also a unique result of an agent learning. One agent's pattern should not be used by another agent directly. The other agent does not have the same pattern generating context as the initiate agent and the networks of recognizing trust for both agents are also formed in very different ways. The reputation can be simple messages that reflect recommendations of the host agent toward the target agent. When the agent requiring reputations receives the messages, the messages also become part of the input data of the trust learning algorithm. Such design is good because even the initiate agent send biased reputation; the receiver agent has the ability to learn it and adapt to the "lying" reputation. Figure 3.7 illustrates the process of generating, transmitting and receiving reputations.

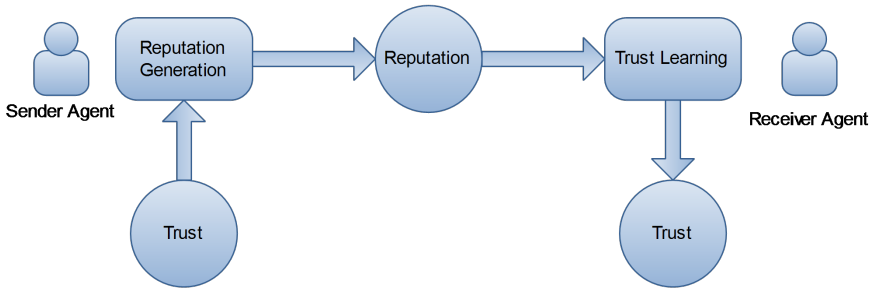

Figure 3.7: Reputation generation - The process of generating, transmitting and receiving reputations.

Propagation is the major feature of the computational reputation. It lets the reputation transmit to two or more agents and it also makes the computational trust become a social influential factor for the virtual society. The agents in the propagating path

creates a chain of reputation propagation. The length of the chain equals the number of agents in that chain. The chain cannot be infinite due to the physical limitations and it is also bounded by the size of the virtual society. The maximum propagating distance is the length of the reputation chain. The distance can be set as a fixed length so that all the agents in the radius of the length are included and propagated. The distance can also be set to be dynamic. The concept of decay will help determine the dynamic distance.

There are three types of propagations: amplifying propagation, unaltered propagation and decaying propagation. In the amplifying propagation, the reputation is amplified a bit through every agent in the propagating chain. It is not an ideal way of propagation. In human society, rumor spreading is an example of such type of propagation (Moreno et al., 2004). Such amplification is not controlled and sometimes it is negative and unpredictable. In a virtual society, amplifying propagation is superfluous or even harmful. So it should be avoided instead of being taken into consideration. In the unaltered propagation, the reputation remains unchanged regardless how many agents it has been passed through. The computer based agents are capable of memorizing all the related information. They do not suffer from aging and forgetfulness like human being, the reputation can reach the receiver agent intact. This is a good feature for keeping the accuracy and reliability of the reputations. In the decaying propagation, the reputation simulates the decaying phenomenon that happens in human society, the magnitude of the reputation decay along the transmitting chain. Each agent receives a shrunken reputation from its upper node. Decaying can keep the length of the propagation chain down dynamically. When the magnitude of reputation shrinks to zero at some agent, the chain is ended and the propagating is over.

From the initiative of the propagating reputations, there are two types of propagation: active propagation and passive propagation. In active propagation, an agent initiates the reputation propagation to its neighboring agents after it finishes an interaction with another agent. The reputation network is updated once an interaction happens between any two agents. For multi-agent system, the advantage of such mechanism is the timely updating of reputations. The disadvantage is that it may lead to big burden on performance. In the passive propagation, the agent does not initiate any active propagation unless it is inquired to do so. The advantages of passive propagation

are that the system can avoid redundant propagation and improve performance. The disadvantage is that the update of reputations lags behind the interactions.

3.3 Model framework

After the analysis and discussions about the trust and the reputation, it is time to have a global framework designed for the computational trust and reputation. The frameworks design needs to reflect the design considerations in the previous section. Each facet of the considerations can be transformed into a mechanism design in the framework. The framework then becomes a systematic composition that is composed by several components.

3.3.1 Model structure

Figure 3.8 illustrates structure of the computational trust and reputation framework. The outside rectangle delimits the bounds between the internal mechanisms of an autonomous host agent and the external environment that the other agents participate. The inside rectangles represent the processes for specific calculations or algorithms. The cycles represent data, information or results fed into the agent or sent out from the agent. The rounded rectangles represent the processed information, knowledge or patterns that are generated by the host agent. The actors represent the host agent and the receiver agent who receives reputations from the host agent. The arrows show the direction in which the information flows.

The framework is constructed with five major mechanisms (components). Inside the host agent: Trust learning mechanism, Trust estimation mechanism, decision making mechanism and reputation generation mechanism. Outside the host agent: the reputation propagation mechanism. There are no priorities for these mechanisms. Each mechanism can work independently, that is, each one can work parallel to one another. The direction of the arrows indicate where the information about transactions flow to which component in the framework. They also indicate the order that the transactional information is processed.

Thus the framework works in three ways simultaneously.

- Continuous trust learning: The first way is continuous trust learning. An agent perceives and extracts the transaction or interaction related data as input to its

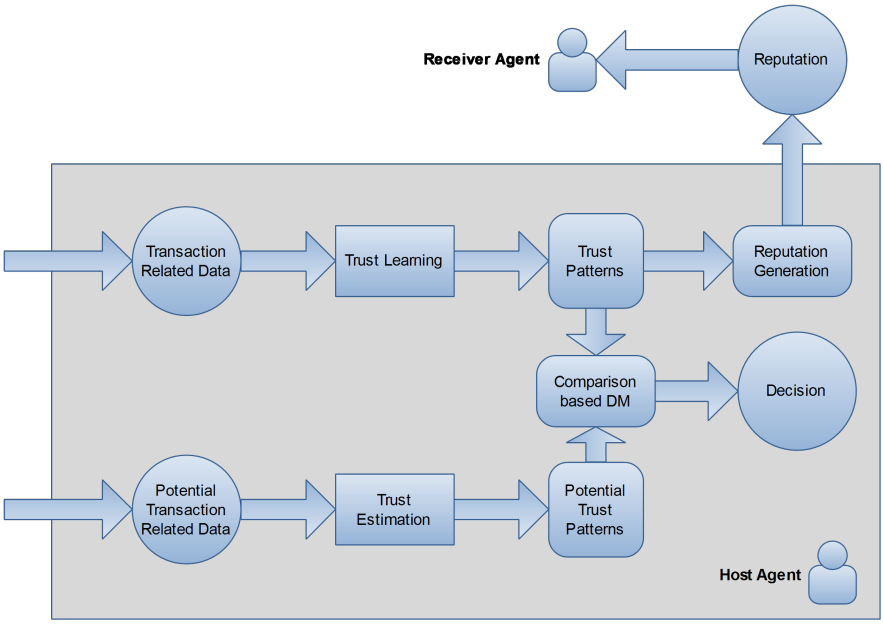

Figure 3.8: Framework - Framework of Computational Trust and Reputation

learning algorithm. The algorithm processes the input data and generates the computational trust patterns. Those patterns are stored for future queries and comparisons. The continuous trust learning is the regular learning accompanied with any agent interactions. Each interaction leads to a repeated learning process. This learning is also the most frequent operation among the mechanisms of the framework.

- Trust estimation and trustworthiness decision making: The second way is trust estimation and trustworthy decision making. An agent prepares to have an interaction with another agent. Before the interaction actually occurs, the agent collects domain related information about the target agent. According to the potential interaction related data, the algorithm makes estimation and produces an estimated trust pattern. This estimated pattern is compared with the existing patterns that generated in the previous interactions. The result of the comparison is then an indicator of the trustworthiness or untrustworthiness toward the target. It helps the agent make a decision whether to trust a specific target agent to start an interaction or transaction.

- Reputation generation and propagation: The third way is reputation generation. When an agent initiates reputation propagation or it is required to do so, the historical patterns of the target agent are found out and compared with the memorized patterns. The result of comparison is also indicator of the trustworthiness of the target agent. This result is transformed into a form of message that is fit for propagation and then is sent out to the receiver agents. The propagation mechanism controls the active/passive transmission of the reputation, the path of reputation transmission, the distance of the transmitting, and the decaying rate of the transmission.

The next sections will overview the critical design in each of the mechanisms. The order of explanation is: Section 4.2 explains the design of the trust learning mechanism, section 4.3 explains the design of trust estimation and trustworthy decision making mechanism, and section 4.4 explains the design of reputation generation and propagation mechanism.

3.3.2 Trust learning mechanism

In this thesis, the computational trust is deemed as patterns learned from agent interactions. Section 3.1 proposes to use neural networks as the basic algorithm that generates the trust patterns. Also according to the discussion in Section 3.1, the learning of trust patterns is emerged and unsupervised. This creates some preconditions for choosing appropriate specific algorithm in the alternatives of neural networks. The thesis chooses Self Organizing Map (SOM) as the basis of the computational trust learning algorithm. The researcher improves the traditional SOM and proposes a SOM based Trust Learning Algorithm (STL).

The STL algorithm can be executed in 9 steps: weight initialization, fetching input data, feed forward input, feed forward output, marking patterns, storing patterns, finding neighbors, updating weights and continuation. The task of initialization is to decide the initial weight of the neurons in the network. For each transaction that occurs, an agent receives various dimensions of transaction related results. The information about those results is organized into a vector. Then the feed forward input feeds the input vector to the STL network. After fetching the input vector, the input data is sent to the STL neurons. The major task of Feed forward output is to find out the

winning neuron. As the author introduces in the SOM theory background, finding the largest value of product of weight and input vector is equivalent to find out the smallest distance between the weight vector and the input vector. To make the patterns useful, it is necessary to do a qualitative analysis against it as well. If the interaction is successful, then the pattern generated can be marked as good pattern; if it fails, then the pattern generated can be marked as bad. The drop points that compose the pattern can also be marked as good points or bad points. The perceived patterns can be stored as a two dimensional array. After storing the patterns, the network tends to adjust weights of the neighboring neurons around the center winning neuron. If the neighboring neurons are in the area of the radius, they are excited neighbors. All the neurons in the range of excited neighbors will get positive updating. If there exists multiple transactions that happened successively, the algorithm is executed in a loop.

The STL algorithm produces three types of patterns: transactional trust general, agent trust pattern and macro trust pattern. The transactional trust pattern is the direct product of the STL algorithm. The pattern records the recognized features of a specific transaction that happens between the host agent and its opponent agent. The agent trust pattern is the indirect product of the STL algorithm. It is an aggregation of transactional patterns for one specific agent over a specific duration. It is the agent's memory toward another agent that forms the basis of their interaction experiences. The macro trust pattern is also the indirect product of the STL algorithm. It is an aggregation of the agent patterns for all transactions. The reason of setting a macro trust pattern is that an agent needs a balanced view of trust over different agents and different transactions.

3.3.3 Trustworthiness based decision making

Once the computational trust is recognized, another problem to be solved is how an agent makes a trustworthy decision which will lead to a future transaction or a future cooperation based upon the recognized trust patterns. The thesis proposes a "Trustworthiness Estimation Algorithm" to solve the problem. The algorithm absorbs neural network component of the STL algorithm to do pre-recognition jobs and adds a search and comparison components to do estimation and recommendation jobs.

The algorithms can be expressed as having 6 steps: fetch input data, feed forward input, feed forward output, searching patterns, comparing patterns and giving recom-

mendations. The process of fetching input data is as same as the first step in the STL algorithm. The input data are fed to the network in the form of input vector with various dimensions. The feed forward input in the estimation algorithm is as same as the feed forward input in the STL algorithm. The process of feed forward output in the estimation algorithm is as same as the process of feed forward output in the STL algorithm. But the result is processed in a different way. Once the predicted pattern is generated, a corresponding pattern should be retrieved from the agent pattern library. The identity of the target agent becomes the key to searching. Comparing patterns is the most critical step in the estimation algorithm. The basic principle of comparison is to put two patterns together and measure how big the differences are. There are mainly two types of difference: different drop points and different size of drop points. After calculating the result of the comparison between the macro trust pattern and the predicted trust pattern, the result can be an indicator of the degree of trustworthiness.

Here is a simple example of pattern comparison. The following matrices are used to represent different patterns. Matrix 1 represent the macro pattern:

$$\begin{pmatrix} 0 & -3 & 1 \\ 3 & 10 & 0 \\ 6 & -5 & 0 \end{pmatrix}$$

matrix 2 represent the estimated trust pattern 1:

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

and matrix 3 represent the estimated trust pattern 2:

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

In the macro pattern, the value as the element of the matrix represents the sum of the drop points which is the result of trust recognition after each round of transactions. In the estimated trust pattern, the value as the element of the matrices represent the exact drop point from the trust estimation before the transaction actually happening. Compare matrix 1 and matrix 2, the drop point of matrix 2 matches the point in matrix 1 with larger positive value which means better chance to be trustworthy. Instead,

compare matrix 1 and matrix 3, the drop point of matrix 3 matches the point with negative value that represents less trustworthy.

3.3.4 Reputation generation and propagation

The basic trust pattern for an agent is a matrix with two types of drop points. The size of the success point or failure point is proportional to the times of successful interactions or failure interactions. The size difference provides a foundation about how the agent believes the trustworthiness is. The history of past interaction can also be recorded using the same style of pattern like the general trust pattern. The only difference is that the general pattern is design to contain all the drop points in all previous interactions. The drop points appeared in the specified pattern must be accumulated to the points on the general pattern. A reasonable reputation should consider both the general pattern and the specified pattern so that the individual performance can be compared with the general performance. The calculation of the difference should be put into a mathematical framework so that the result of the calculation is a numerical value which can be propagated along the network, accepted and absorbed by the inquiry agent.

The initiative agent sends the reputation value to the inquiry agent. The reputation receiving agent should also take the sender's reputation into account. Propagating the reputation through inquiry method may form a chain that connects multiple agents which forward or respond to the inquiry request. The thesis introduces a variable called "decaying coefficient", which can help adjust the degree of decaying speed. The coefficient could be moderate so that the distance of propagation is confined to reasonable clusters. When an agent accepts the reputation, it needs to update its trust pattern against specific agent using this reputation. The updating method is to simply put the reputation as the input data for the STL learning algorithm.

The thesis designs three forms of reputation propagation mechanisms: point-to-point based inquiry, broadcasting based propagation and observer based propagation. The point-to-point based Inquiry is the simplest and most effective way of the reputation propagation. In order to have some knowledge of the target agent, an agent can send a request to its own acquaint agents. The broadcasting propagation can cover wider propagation scope in a short time. Once the initiative agent has interacted with some specific target agent, it will generate a reputation reference and send this reference to all the agents in its domain or society. In the observer propagation, agents can be

categorized into four roles: subject, target, subscriber and un-subscriber. The subject is an agent being watched by the other agent. Its behavior and its interaction with the other agents are interested by other agents. The key to design a subject agent is that it holds a subscriber list that contains all the agents' identities who are interested in the subject. A target agent is the agent that interacts with the subject agent. The result of their interaction will be propagated as a reputation reference.

Point-to-point based inquiry is appropriate for scenario that agents have clear list of the other agents in the virtual society. The efficiency of propagation is high because there is no redundant communication channel between agents who do not involve in the interaction. Broadcasting propagation is good for a simple scenario with limited number of agents and there is a necessity to let every agent in the society to know what happens. The efficiency of propagation is low because large bandwidth is wasted to build communications with every agent. Observer propagation is good for large scale virtual society like electronic commerce market. An agent only propagate reputation to those who have registered in its subscriber list. The efficiency of propagation is between the broadcasting propagation and point-to-point based inquiry.

3.4 Summary

This chapter first analyzes the concept of the computational trust and computational reputation. It figures out six core features of the computational trust, and explains the influences of each feature. This chapter also compares the computational trust with the human trust thus the reader can understand the origin of the computational trust and have insight about how computational trust is superior to the human trust. Then the chapter designs a systematic framework for the neural trust model which includes mechanisms and algorithms such as trust learning, trust estimation and reputation generation, and reputation propagation. The relationships among components are introduced and each component is also described briefly.

4

Trust Learning and Estimation

4.1 Introduction

In the neural trust model constructed in the previous chapter, it is obvious that the component used for learning and making decisions is the most important component. So the algorithms inside the component are major topics of this chapter. The requirement of the model is to make the agents learn the experiences of their past interaction. This determines that the algorithm must be a learning algorithm. Additionally, the pattern produced by the algorithm should be easy to memorize, store and reuse.

As an unsupervised learning scheme, Self Organizing Map is the best choice for recognizing interaction data and find out features of the interaction. This chapter discusses the rationale of choosing SOM, and introduces the SOM theory in detail and describes the standard algorithm.

Based on SOM, two new algorithms are proposed: SOM based trust learning (STL) algorithm, and SOM based trust estimation (STE) algorithm. STL is designed as learning the trust from the interactions while STE is design to learn the potential interaction data and compare the related pattern with the STL generated pattern so that it can decide whether the specific agent is trustworthy. The key results of both algorithms and the patterns are categorized as three types and each type get explained thoroughly.

The chapter contains six sections. The organization of the content is from general to specific, from basic to advance. Section 2 describes the theory about Self Organizing Map, introduces its features and depicts the process and algorithm. Section 3 proposes

SOM based trust learning (STL) algorithm and describes this algorithm step by step. Section 4 differentiates the different patterns generated from the STL algorithm and discusses their characteristics and possible usages. Section 5 proposes SOM based trust estimation (STE) algorithm and describes this algorithm step by step. The section also presents the mechanism of decision making based on the results of STE algorithm. Section 6 summarized the chapter.

4.2 Theory of Self Organizing Map

4.2.1 Self Organizing Map

Self organizing map (SOM) is one type of neural networks based on competitive learning (Kohonen, 1998). The features of SOM include: approximation of input spaces, topologically ordering, density matching and feature selection (Haykin, 1999). The belief of trust is frequently mutating and updating, the chosen algorithm should be highly adaptive. This is what SOM is capable of. The belief of trust is a continuous accumulation of information. The output of SOM is compact and complete enough to reflect every piece of changes. Also, the output of SOM can reveal obvious characteristics of the trust belief. It is easy for a computer program to discover the belief of trust which comes from multiple sources.

The SOM can approximate the input space. The statistical characteristics can be extracted from the input data. The result map reveals an approximation of the input space through the synaptic weight of the output space. The SOM can discover the topological ordering of the input data. The location of the excited neurons construct a spatial pattern that represent the geometric characteristics of the input space. Because the neighboring neurons are also excited, the zone that reflects the input space has better chances to be selected. This results in a good density matching. Thus, the features hided in the input space can be found through discovering the spacial distribution of the neuron groups.

The basic process of SOM is made up by three processes: competitive process, cooperative process and adaptation process. After initializing the synaptic weights of the network, in the competitive process, the data of the input pattern are provided to the discrimination function of the network, for each input pattern at each neuron, there will be a value of the discrimination function. The highest value gained neuron will

be the winner of the competition. In the cooperative process, the neighbor function decides which neighbor neurons around the winning neuron are excited as well. In the adaptation process, the synaptic weights are adjusted so that the similar pattern leading to the winning neuron will be enhanced.

4.2.2 Learning process

Based upon the traditional SOM algorithm, the trust learning process is composed by three processes: competitive process, cooperative process and adaptive process. In the competitive process, the discriminative function will help the network identify the winning neuron which produces the largest value of the discriminative function; in the cooperative process, according to the winning neuron's topological location, the neighboring neurons are calculated to be excited; in the adaptive process, the excited neurons and the winning neuron can adjusted their weight so that the ability of recognizing the next similar patterns are gradually enhanced.

Mathematical or formal description:

- Competitive process: Let n denotes the dimensions of the input data to the network. Then the input data can be described using a vector in the form of a transpose matrix.

  $$x = [x_1, x_2, \ldots, x_n]^T \quad n \geq 1$$

  The dimension of the trust is already discussed in Chapter 3. Theoretically speaking, the dimension should not be limited in its size. But the reality is different. The size is confined by the physical capability of the computer which holds the agents. It is impossible to increase the dimension indefinitely and it is also not necessary to increase the dimension indefinitely. In this chapter, the dimension is mainly focused in a simulated way, that is, only up to 7 dimensions are used to illustrate how the agent trust is recognized.

  For each neuron in the network, there is a corresponding synaptic weight. The weight of each neuron is represented as w. The number of weights is equals to the number of dimensions (n). So the weights can be depicted as:

  $$\omega = \omega_{j1}, \omega_{j2}, \ldots, \omega_{jn} \quad j = 1, 2, \ldots, l$$

l is the total number of the neurons in the SOM network. In order to find out the winning neuron from the network, it is necessary to look for the biggest value of the inner product of ω and x. Because the same threshold is negative bias and is applied to all the neurons. When the largest inner product $\omega_j^T x$ is selected, the corresponding neuron can be determined as the center of excited neurons. This is equal to find out the smallest Euclidean distance between vectors ω_j and x. If i(x) is used to represent the distance, then in order to get the biggest inner product $\omega_j^T x$, the $i(x)$ should be:

$$i(x) = \text{argmin} \|x - w_j\| \quad j \in A$$

A represents all the neurons appeared in the network. $i(x)$ is the neuron that hold the biggest inner product value. The index of this $i(x)$ can determine the topological position of the excited neuron. The neuron is called winning neuron or best matching neuron in SOM algorithm.

- Cooperative process: The key operation of the cooperative process is to find out the neighboring neurons which should be excited with the winning neuron. The method is to use the function to calculate whether the neighboring neurons are in the excited range of the winning neuron. There are three main types of neighboring functions: Mexican hat function, Stovepipe hat function and Chef hat function (E. and Yuhui, 2007). The most often used function is the Mexican hat function (it is formally called Gaussian function (E. and Yuhui, 2007)). Let $h_(j, i(x))$ represents the set of neighboring neurons that will be excited. Let $d_(j, i)$ represents the distance between the winning neuron and the neighboring neuron. The rule is simple, the distance decides whether the neurons get excited or not. It is obvious that $d_(j, i)$ is inverse proportion to $h_(j, i(x))$. When the distance $d_(j, i) = 0$, the $h_(j, i(x))$ will get the maximum value at the winning neuron. The distance $d_(j, i)$ can be infinitely large, that is, there can be infinite number of neighboring neuron, but the $h_(j, i(x))$ tends to be zero.

- Adaptive process: After finding out the excited neighboring neurons around the centered winning neuron, it is time to update the weights of those excited neurons. This is the most critical part of self organizing. The weight change can be

calculated as follows:

$$\Delta\omega_{ji} = \eta(t)(\alpha_{ki} - \omega_{ji})$$

$\eta(t)$ is a decreasing function of time which represents the decaying of the weight changes. $(\alpha_{ki} - \omega_{ji})$ is the smallest Euclidean distance of the input vector and weight vector. $\Delta\omega_{ji}$ should be added to the weight so that the winning neuron can be rewarded with more weight to reflect the classification of the input vectors. So the weight can be calculated as follow:

$$\omega_{ji}(t+1) = \omega_{ji}(t) + h_{(j,i(x))}(t)\eta(t)(\alpha, i_{ki} - \omega_{ji})$$

In this formula, $h_{(j,i(x))}(t)$ is the neighboring function as mentioned above. $\omega_{ji}(t)$ is the old weight for the particular neuron. $\omega_{ji}(t+1)$ is the new weight after combining the positive feedback. All the neurons in the range of excited neighbors will get positive updating.

4.3 SOM based Trust Learning Algorithm (STL)

In order to let agents fully exploit their previous experiences of interaction, it is necessary to make agents learn trust from their old experiences. The thesis proposes an algorithm called "SOM based Trust Learning Algorithm (STL)" to help agents recognize the features related to trust through their repeated transactions. The algorithm is also the prelude of decision making toward trustworthiness in the next stage. The algorithm can be executed in 9 steps: weight initialization, fetching input data, feedforward input, feedforward output, marking patterns, storing patterns, finding neighbors, updating weights and continuation. The following pseudo code describes the STL algorithm:

```
1.   Weight initialization;
2.   While(transaction occurs;){
3.   Fetching input data;
4.   Feed forward input;
5.   Feed forward output;
6.   Marking pattern;
7.   If (review result is good)
{ Marking patterns as good;}
else
{ Marking pattern as bad;}
8.   Storing patterns;
9.   Finding neighbors;
10. Updating weights;
11. Next round;
}
```

Figure 4.2 uses a process graph to depict these steps.

1. Weight initialization: The task of initialization is to decide the initial weight of the neurons in the network. In the STR algorithm, each weight is assigned a random weight as its initial weight. The random value is in the range of 0.4 to 0.6. The bias of initial weight does influence the performance of the recognition. However, in an unsupervised trust learning, trust dimensions

are provided to the STR algorithm one at a time when an agent transact with another agent. The STR algorithm tends to form a stable weight value through many transactions in a relatively long duration. So the initial bias on weight value only has minor influence on this prolonged process.

2. Fetching input data: For each transaction that happens, an agent receives various dimensions of transaction related results. The information about those results is organized into a vector which has been mentioned in chapter 4. For example, an input vector with five dimensions reputation, price, appearance, quality, delivery will look like 1, 1, 0, 1, 0. Thus the input data to the STL algorithm is actually a matrix which contains many rows of transaction related information. Such matrix is not prepared beforehand. It is collected by the agent after every transaction. Each transaction adds one row of information into the matrix and the agent fetch this row to learn and then generate the trust patterns. An agent is not trained to understand the pattern, it learned during its lifecycle in continuous business activities. The STL algorithm thus is an unsupervised learning algorithm.

3. Feedforward input: Feedforward input feeds the input vector to the STL network. There are only two layers in the STL algorithm: the input layer and the output layer. Or they can be called the sensor layer and the recognition layer. The input layer is responsible for receiving the data from the input vector. The number of nodes in this layer depends on the number of dimensions in the input vector. One input node corresponds to one dimension in the input vector. The number of nodes is dynamic. The more sophisticated the transaction is, the more nodes might be needed to help sense more information. Chapter 4 has discussed how the dimension is defined and how two different agents achieve agreement on the meaning of the transaction. After fetching the input vector, the input data are sent to the STL neurons.

4. Feedforward output: The major task of Feedforward output is to find out the winning neuron. As the author introduces in the SOM theory background, finding the largest value of product of weight and input vector is

equivalent to find out the smallest distance between the weight vector and the input vector. The output neuron with the minimum Euclidean distance is the winning neuron which stands for the classification that the input vector belongs to. Thus the activation function for the Feedfoward output is:

$$d = \sqrt{\sum_{i=1}^{n} (\alpha_{ki} - \omega_{ji})^2}$$

d represents Euclidean distance, n represents the number of neurons in the output layer, α_{ki} represents the input vector and ω_{ji} represents the synaptic weight of the neuron. The distance is calculated for each neuron and all the distances are compared to find out the smallest one. Then the neuron which has the smallest distance is set to the winning neuron.

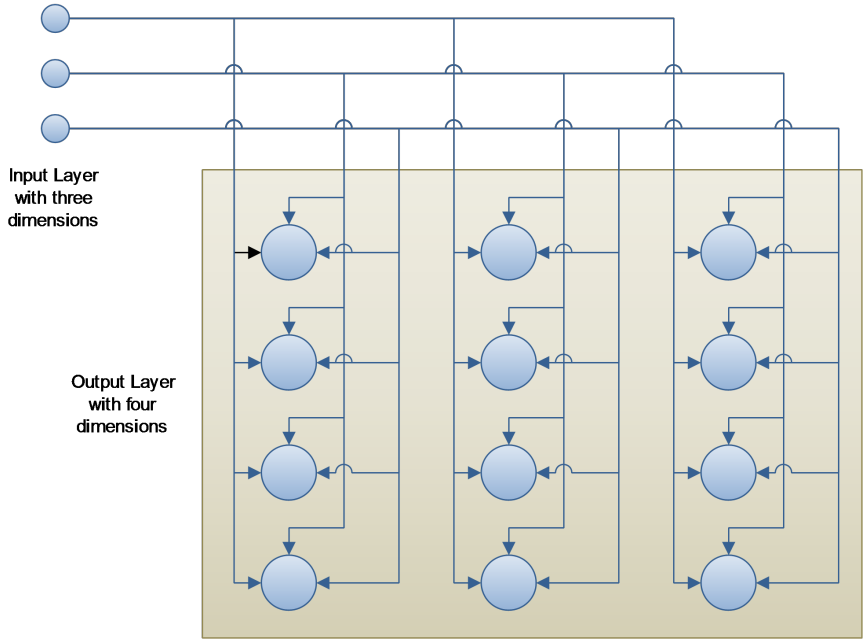

Figure 4.1: Map Structure - The structure of the SOM network

5. Marking patterns: It is nonsense to talk about the goodness or badness of the pattern recognized. A pattern is just a neutral concept. Only when agents or people endow their interests to the pattern, the pattern then become valuable and useful. Like the natural concept "water", people think water is good because water help them survive, irrigate their farm, and

present them with beautiful scenery. To make pattern useful, it is necessary to do a qualitative analysis against it as well. The pattern itself is not analyzable. The analyzable target is the transaction which provides the input information and generates the pattern. It is not important to know the details about the transaction. The most straightforward question can be ask is whether this transaction is successful or has failed. If it is successful, then the pattern generated can be marked as good pattern; if it has failed, then the pattern generated can be marked as bad. Furthermore, the drop points that compose the pattern can also be marked as good points or bad points. Now the pattern is useful when an agent enters a new transaction and needs to estimate the trustworthiness of the new opponent agent. They only need to compare the difference between the pattern from the new transaction and the memorized good patterns. In the next section, the thesis will discuss the trustworthiness estimation algorithm in details.

6. Storing patterns: The thesis aims to find a way to estimate the trustworthiness of agents and transactions. Historical patterns are thus the source of estimation and they should be stored in a proper way so that the patterns can be retrieved conveniently in the next time. The pattern is a matrix which is composed by blank point and drop points. It can be stored as a two dimensional array. The following matrix is an example of the array: b represents blank point and d represents drop points.

$$\begin{bmatrix} b & d & b \\ d & b & b \\ b & d & b \end{bmatrix}$$

For every new transaction with a specific agent, there will be a new pattern generated accordingly. For an agent in a transaction, there is a pattern stored in the following relational form: For a specific agent who has more than one time interaction with the host agent, there is an accumulated general pattern stored in the following relational form: For the host agent, there is a meta-level pattern which can help agent deal with new transaction with new agent who has never met before. Such meta-level pattern can be called macro pattern and can be stored in the following relational form:

Agent id	Transaction id	Pattern generated	Marking
Agent-xxx	10645571	$\begin{bmatrix} b & d & b \\ d & b & b \\ b & d & b \end{bmatrix}$	Bad

Table 4.1: Example for storing transactional patterns - Example of the pattern storage for each transaction.

Agent id	General Pattern	Marking
Agent-xxx	$\begin{bmatrix} b & d & b \\ d & b & b \\ b & d & b \end{bmatrix}$	Good

Table 4.2: Example for storing general patterns - Example of the pattern storage for general patterns.

Marking	Macro Pattern
Good	$\begin{bmatrix} b & d & b \\ d & b & b \\ b & d & b \end{bmatrix}$

Table 4.3: Example for storing macro patterns - Example of the pattern storage for macro patterns.

7. Finding neighbors: The most obvious difference between SOM network and the other type of neural networks is that the SOM network tends to adjust weights of the neighboring neurons around the center winning neuron. This is a lesson learned from the mechanism of the human brain. In the human brain, there is evidence that firing a biological neuron can also make the adjacent neurons to be excited (Ritter, 1999). The neighboring function is the key to control how many neighbors can be excited. The theory background has introduced three types of neighboring functions: Gaussian function, Stovepipe hat function and Chef hat function (E. and Yuhui, 2007). In the STL algorithm, the simplest method is used, that is, the Chef hat function is used as the neighboring function. The Chef hat function only sets the radius around the winning neuron. If the neighboring neurons are in the area of the radius, they are excited neighbors. Otherwise, the outside neurons are never influenced by the weight updating.

8. Updating weights: Updating the synaptic weights of the winning neuron and its neighbors is how the network adapts to the ever changing environment. The equation of updating the weights has been mentioned in the theoretical background.

$$\omega_{ji}(t+1) = \omega_{ji}(t) + h_{j,i(x)}(t)\eta(t)(\alpha_{ki} - \omega_{ji})$$

   $h_{j,i(x)}(t)$ is the neighboring function as mentioned above. $\omega_{ji}(t)$ is the old weight for the particular neuron. $\omega_{ji}(t+1)$ is the new weight after combining the positive feedback. All the neurons in the range of excited neighbors will get positive updating.

9. Continuation: If there is only one transaction that happens, the algorithm can be stopped. The agent can wait for another round of transaction and wait for another vector of information to be fed. If there exists multiple transactions that happens successively, the algorithm is executed in a loop. Continue with step 2 until there is no more input vector to be fed.

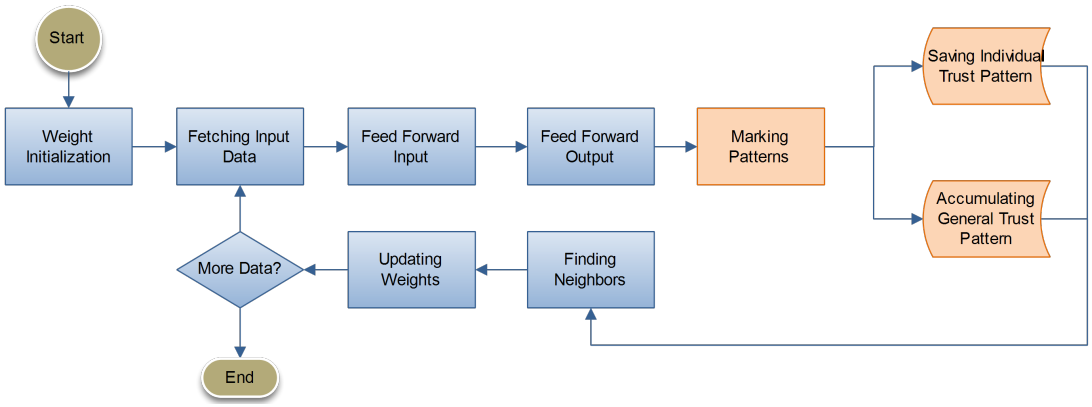

Figure 4.2: STR Algorithm - The algorithm of SOM based trust learning

4.4 Features of the generated trust patterns

4.4.1 Three types of trust patterns

The STL algorithm produces three types of patterns: transactional trust general, agent trust pattern and macro trust pattern. Their abstraction level is meta-level for all the agents in all transactions, general level for a specific agent in all previous transactions and concrete level for a specific agent in a specific transaction. The hierarchy of the three levels is depicted in Figure 4.3. A macro pattern is an aggregation of many agent patterns; an agent pattern is an aggregation of many transactional patterns. The network of STL algorithm only directly generates the lowest level pattern, the transactional pattern. The other two types of patterns are created according to the transactional patterns indirectly. All three types of patterns are results of learning activities while only the higher two types, macro pattern and agent pattern, are used in the trustworthiness estimation activities.

4.4.2 Transactional trust pattern

The transactional trust pattern is the direct product of the STL algorithm. The pattern recorded the recognized features of a specific transaction that happens between the host agent and its opponent agent. Once the pattern generated is marked as good or bad, the pattern becomes the source material of the above two level of patterns. The transactional pattern is stored in a matrix (two-dimensional array). Every point in the matrix represents a potential feature of the target agent. For each transaction, after

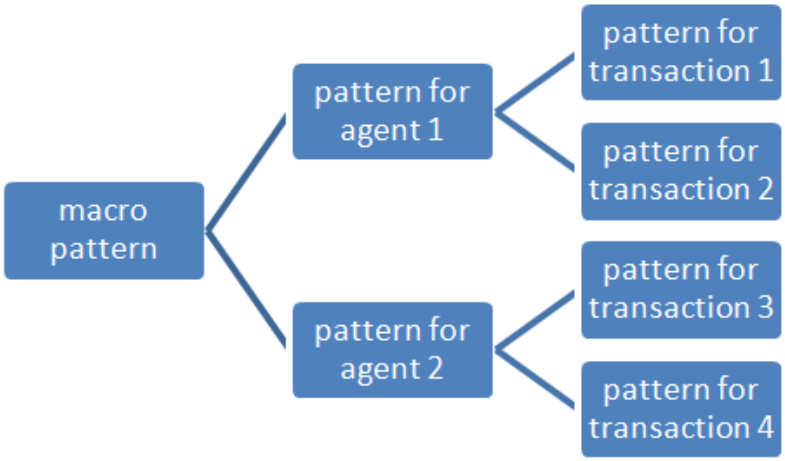

Figure 4.3: Pattern types - Hierarchy among three types of patterns

the input data fed into the network, a pattern with a drop point is created. The drop point is actually the winning neuron of the algorithm. Its location is kept intact and recorded with the pattern. Figure 4.4 shows an example of the transactional pattern. The example presents an output layer with 3 dimensions and 9 neurons. The red point in the graph represents the drop point or the winning neuron.

4.4.3 Agent trust pattern

The agent trust pattern is the indirect product of the STL algorithm. It is an aggregation of transactional patterns for one specific agent over a specific duration. It is the agent's memory toward another agent based on their interaction experiences. The agent trust pattern is useful when the host agent is inquired by other agents about some specific agent's reputation. The agent compares the agent trust pattern with its macro trust pattern and then decides to send a reputation to the inquiring agent. The details of reputation generation and propagation are discussed in Chapter 6. Because the transactional patterns have good or bad marking, the agent trust patterns have good or bad marking as well. Figure 4.5 illustrates an agent trust pattern marking as good. In this figure, the red point represents the drop points of previous transactions. The size of the point states that the same drop points has appeared in the same location for several times. Actually, the agent trust pattern can be viewed as a overlapping of many transactional patterns.

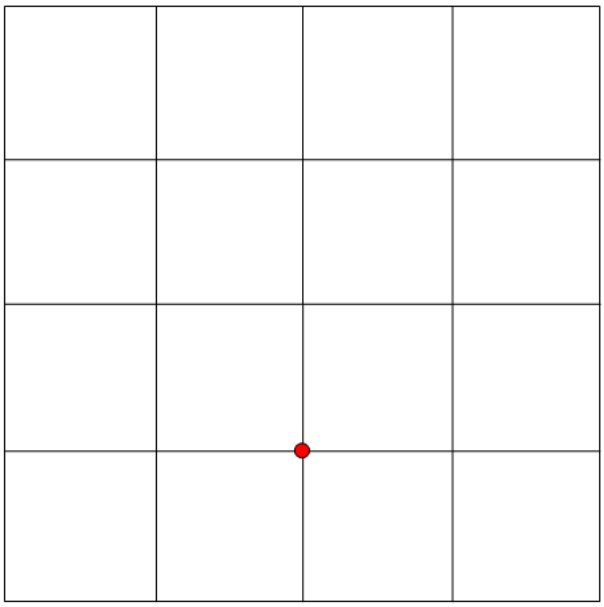

Figure 4.4: Transactional pattern - An example of the transactional pattern

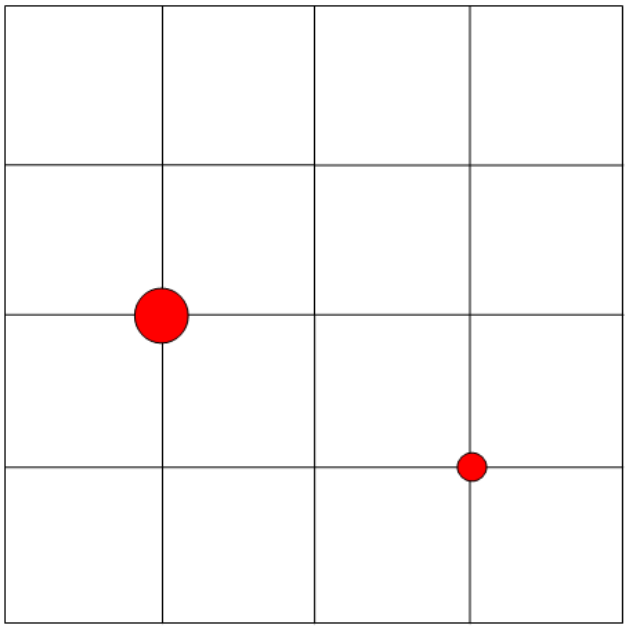

Figure 4.5: Agent trust pattern - An example of the agent trust pattern

4.4.4 Macro trust pattern

The macro trust pattern is also the indirect product of the STL algorithm. It is an aggregation of the agent patterns for all transactions. The reason of setting a macro trust pattern is that an agent needs a balanced view of trust over different agents and different transactions. If there are no macro trust patterns, then the agent lacks a benchmark to measure how big the differences are among agents and among transactions. This is important especially when the host agent needs to send reputation to its partner agent. Different from the transactional trust patterns and the agent trust patterns, the macro pattern is only one integral pattern instead of division based on good or bad marking. It is an aggregation of both good agent trust patterns and bad trust patterns. It reveals the whole trust map of the host agent and builds a foundation of the reputation network. Figure 4.6 illustrates an example of the macro trust pattern. In this figure, the red points represent the good drop points that are led by successful transactions and the black points represent the bad points that are led by the failed transactions. The size of the points is the result of aggregation of many agents' trust patterns. If unfortunately, the good drop point coincides with the bad drop point, they counteract each other and carry out a subtraction in effect.

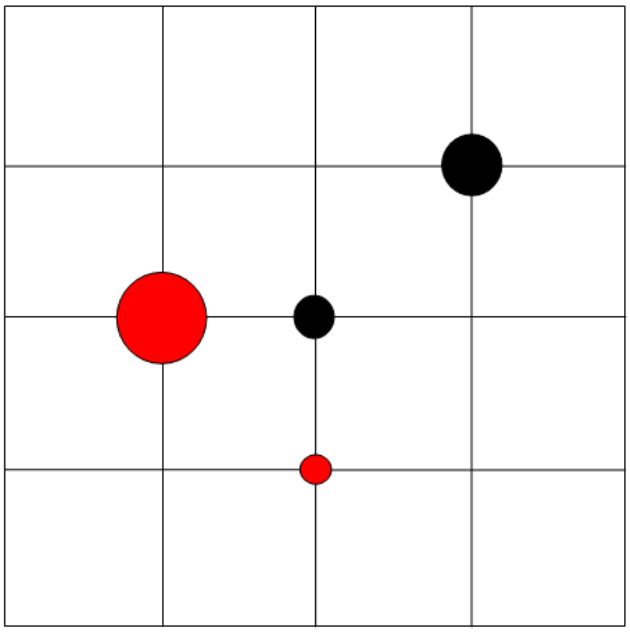

Figure 4.6: Macro trust pattern - An example of the macro trust pattern

4.5 SOM based Trust Estimation Algorithm (STE)

The STL algorithm help agents find trust patterns through process the previous transaction experiences. It also helps agents store those patterns into its memory as three abstraction levels. However, this only solves the first fold of the problem: seek the trust patterns for the agents. The second fold of the problem is how an agent makes a trustworthy decision which will lead to a future transaction or a future cooperation based upon the recognized trust patterns. The thesis proposes a "Trustworthiness Estimation Algorithm" to solve the latter problem. The algorithm absorbs neural network component of the STL algorithm to do pre-recognition jobs, and adds a search and comparison components to do estimation and recommendation jobs. The algorithms can be expressed as 6 steps: fetch input data, feedforward input, feedforward output, searching patterns, comparing patterns and giving recommendations. The following pseudo code depicts the STE algorithm:

```
1. While(meet new potential transaction;){
2.     Fetch input data;
3.     Feed forward input;
4.     Feed forward output (transaction pattern);
5.     Searching  macro patterns;
6.     Comparing patterns (macro pattern vs. transaction pattern);
7.     Generating recommendations;
8.     Next round;
}
```

Figure 4.7 illustrates the steps of the estimation algorithm.

1. Fetch input data: The process of fetching input data is as same as the first step in the STL algorithm. The input data are fed to the network in the form of input vector with various dimensions. But the type of input data being used is different. In the STL algorithm, the input data are mainly information related to a previous transaction. A host agent gains experiences from analyzing its historical data. The pattern referring to the transaction and the target agent is also generated and stored. However in the estimation algorithm, the purpose of the estimation is to predict whether a target agent can be trusted to carry out a

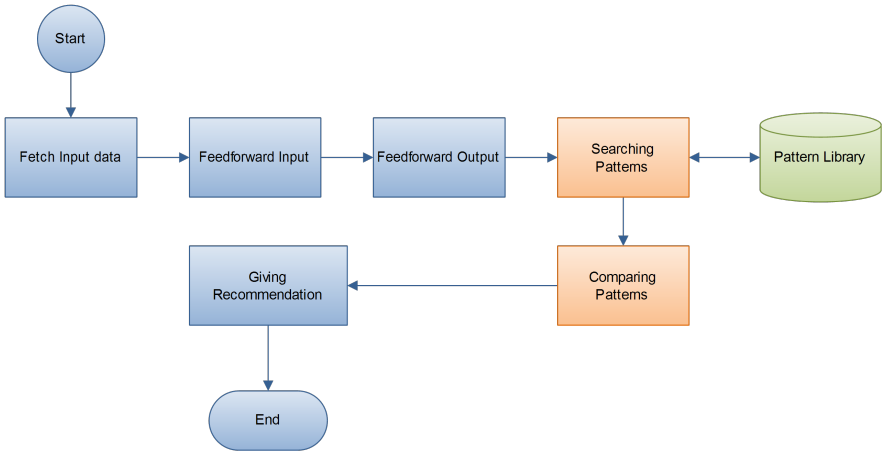

Figure 4.7: STE algorithm - The process of the trust estimation algorithm

transaction with the host agent. The data used as the input vector is not historical information from old transaction. The data used in the estimation algorithm is observed data before any the transaction actually happens. The observed data also follows the format of the input vector, that is, it should have the same format and dimensions as the real transactional input vector. Otherwise it would be impossible to compare the transaction generated patterns and the estimation generated patterns. For example, if there is a book to be sold through automatic agent transaction, the information to describe the book could be organized from 5 dimensions: reputation, price, quality, appearance, delivery. The reputation can be gained from the other agent (see next chapter about reputation propagation). The price is clearly stated. The quality needs the book to be ranked by some expert's book review or customer's book review like what Amazon has already done. Appearance and delivery also needs a third party to give them a rank so that the agents can retrieve it directly. The mechanism of ranking is not in the scope of the thesis; the reader can find more information about ranking through (Koehn, 2003).

2. Feedforward input:The feedforward input in the estimation algorithm is as same as the feedforward input in the STL algorithm. They are both based on the SOM algorithm. Their structure is also two layers: one layer as input layer and the other layer as output layer.

3. Feedforward output: The process of feedforward output in the estimation algorithm is the same as the process of feedforward output in the STL algorithm. But the result is processed in a different way. In the STL algorithm, the pattern recognized are transformed into three forms: transactional trust pattern, agent trust pattern and macro trust pattern. These three forms become the foundation of the trust estimation. In estimation algorithm, the output pattern is only a predicted transactional trust pattern which aims to be compared with the macro pattern. It is only a temporary intermediary pattern which will be discarded after the estimation. Such discarded pattern will not enter the pattern library and will not form parts of the knowledge about trust.

4. Searching patterns: Once the predicted pattern is generated, a corresponding pattern should be retrieved from the agent pattern library. The identity of the target agent becomes the key to the searching. If there is an id of the target agent in the library, it means that the host agent has already interacted with the target agent before. The agent trust pattern for the target agent and the macro pattern are provided for the next comparison. If the id of the target agent is not in the library , it means that the host agent has never interacted with the target agent before. In this case, only the macro trust pattern is returned.

5. Comparing patterns: Comparing patterns is the most critical step in the estimation algorithm. The basic principle of comparison is to put two patterns together and measure how big the differences are. There are mainly two types of difference: different drop points and different size of drop point. So the comparison must consider both situations. The pattern is stored as a matrix so that the difference between patterns is the difference between matrixes. Let P_e represents the predicted pattern and let P_m represents the macro trust pattern. The difference between these two patterns is:

$$P_d = P_m - P_e$$

Now the job is transformed to measure P_d. Based on P_d, let g_{kl} represents the successful drop points in the predicted trust pattern. Let f_kl represents the failed drop points in the specified trust pattern. Let r_g represent the rate of good drop points in all points in the pattern. Let r_b represents the rate of bad points in

all points in the pattern. Let r_d represents the rate of displaced points in the pattern.

$$r_g = \frac{\Delta g_{kl}}{\sum g_{kl}}$$

$$r_b = \frac{\Delta b_{kl}}{\sum b_{kl}}$$

$$r_d = \frac{\Delta d_{kl}}{\sum d_{kl}}$$

Because the agent has no prior knowledge about what the displaced points are, these points can be good points or bad. In order to state the risk tendency of the agent, a risk factor γ is introduced to adjust the attitude of the agent toward the unknown facts. The value of γ is between -1 and 1. It could also be zero if the agent chooses to omit the displaced point. Let r represents the result of the comparison. So the r is generated according to the following formula:

$$r = r_g - r_b - \gamma r_d \qquad (-1 \leq \gamma \leq 1)$$

$$r = \frac{\Delta g_{kl}}{\sum g_{kl}} - \frac{\Delta b_{kl}}{\sum b_{kl}} - \gamma \frac{\Delta d_{kl}}{\sum d_{kl}} \qquad (k = 1 \ldots i, l = 1 \ldots j, -1 \leq \gamma \leq 1)$$

6. Giving recommendation: After calculating the result r of the comparison between the macro trust pattern and the predicted trust pattern, r can be an indicator of the degree of trustworthiness. There are three situations that r may be:

$$\begin{cases} r < 0 \\ r = 0 \\ r > 0 \end{cases} \tag{4.1}$$

   - In the first situation $r < 0$, there are big gap between the predicted trust pattern and the macro trust pattern for the specific agent. The bigger the absolute value of the r, the smaller the trustworthiness is.
   - In the second situation $r = 0$, this means there is no trustworthiness evaluation. The reason might be lack of the fact supporting trustworthiness evaluation. The agent will not take any action to do further jobs.

- In the third situation $r > 0$, most of the drop points coincide with the good drop points of the macro pattern. In such case, the algorithm encourages the agent to take further action to cooperate, delegate or do some transaction with the target agent.

4.6 Summary

In this chapter, the thesis analyzes the necessity of learning trust in a way of neural network. It also proposes to use self organizing map as the basic algorithm for learning computational trust. The chapter describes the processes of the standard self organizing map and illustrates its core features. Then the chapter proposes the SOM based Trust Learning algorithm and describes the processes of the algorithm in detail. The algorithm improves the traditional SOM algorithm according to the special requirement of trust learning. This chapter also proposes the SOM based Trust Estimation algorithm and depicts its detailed processes. This algorithm is used to estimate the trustworthiness of the target agent before any transactions. It is the core algorithm that help agent make trustworthy or untrustworthy decisions.

5

Reputation Generation and Propagation

5.1 Introduction

Reputation plays an important role in multi-agent system. It is a socialized form of trust which makes agent cooperate with each other and reduces the cost of agents' interaction. In a world with only computational trust, the agent can only perceive its own interactions. Its learned trust pattern can only be used by itself. There is no socialized mechanism to magnify the trustworthiness that has been learned. To introduce reputation is the solution to efficiently exploit the trust patterns. If the NTR algorithm is designed for intelligent agents, then the reputation propagation models and reputation generation mechanism are designed for multi-agent systems. Introducing reputation into multi-agent systems brings many benefits: the agent can greatly extend its range of influence to cover other agents. The agent also can share the interaction experience with others. Such sharing will accelerate the washing out of malevolent agents and increase the possibility of transactions for benevolent agents. The reputation will improve the executive efficiency of agents by avoiding unnecessary communication and transactions. In general, reputation is the key to form a tight coupling agent society.

There is no acknowledged or standard definition for computational reputation. But it is possible to describe it from five facets: interaction experience, intention of propagation, range of propagation, path of propagation, content of reputation. Interaction

experience explains the reputation from the view of information source; intention of propagation explains from the view of agents' motivation; range of propagation explains from the view of spatial consideration; path of propagation explains from the view of network; content of reputation explains from the expression of the reputation.

The author builds three models of reputation propagation. Point-to-point based inquiry allows an initiative agent start an inquiry request to its acquaintance. If the middle agent has intention to transfer the inquiry, then the request can be propagated far from the initiative agent and thus form a reputation network. Broadcasting based propagation is to let agent broadcast its experience about every interaction or transaction so that every other agents in the society can learn what happened.

In this chapter, section 6.2 discusses the difference between human reputation and computational reputation, it also introduces three existing computer based reputation models and analyzes their advantage and disadvantages. Section 6.3 gives a formal definition of computational reputation and describes it from five facets. Section 6.4 describes the design of three reputation propagation models, explains their processes and features in detail. Section 6.5 aims to introduce the reputation generation mechanism and the corresponding mathematical formula; it also explains the design rationale behind the formula in details. Section 6.6 evaluates the propagation models and the reputation generation formula proposed in the previous sections, it uses lab experiment results to prove the validity of the model and discuss some important trade-offs during the design. Section 6.7 is a summary to this chapter.

5.2 Definition

The meaning of reputation in dictionary implicitly refers to the reputation in the human society. There is a big difference between the human reputation and computer based reputation. In addition to the common meaning of the word "reputation" appeared in the dictionary, the author tries to depict the computational reputation, that is, agent reputation from five aspects: interaction experience, intention of propagation, range of propagation, path of propagation, content of propagation. Figure 5.1 illustrates these five aspects.

Interaction experience toward target agent is the source of reputation. An agent first needs to have some interaction experience with the specific target agent. And such

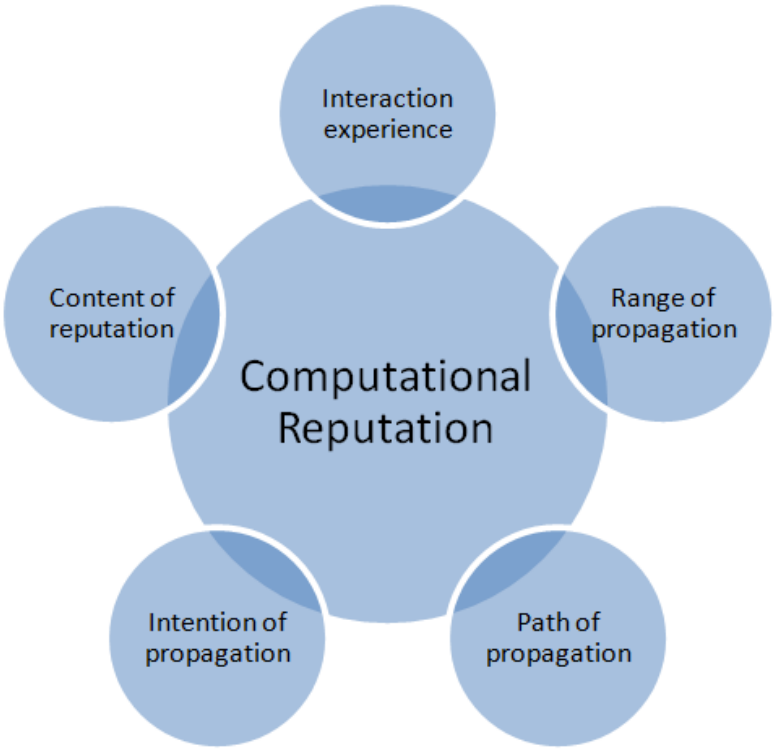

Figure 5.1: Definition of reputation - The dimension of defining a computational reputation

interaction needs to have some results either positive or negative. As the thesis describes in the previous chapter, the agent perceives the result and melts it with its existing pattern toward the target agent. In case of inquiry on the target reputation or in case of broadcasting, the agent will generate a reputation according to the trust pattern and propagate the reputation to some agents or some networks. Trust recognition is the very first step of reputation propagation, only when agents have interacted with some other agents, then they are qualified with the "experience", pattern of trustworthiness, to tell the other agents that agent x is good and agent y is bad.

Intention of propagation is the intent that an agent is willing to propagate its perceived reputation to the other agents. It is like a switch that controls whether the agent is willing to share its interaction experience with its fellow agents over some form of reputation networks. The set of the switch can be used with the goal of an agent to emphasize that an agent is proactive and on its own initiative to participate a network. In this thesis, the author assumes every agent has the initiative to share their interaction experience because the purpose of the thesis is to investigate the mechanism of trust and reputation in a dense network full of active sharing agents.

Range of propagation is the distance that an agent can affect another agent by sending its perceived reputation to. The range here refers to the number of agents that lies between the sending agent and the receiving agent. In a virtual world of computer program, the simplest scenario is to remove the limitation of propagation range, let reputation propagated without any blocking or decaying. Any agent can get undistorted reputation from the remote agent. Agents in such scenario are actually stay in a flat and homogeneous network without layers or clusters which is too na?ve from the real world networks. In the real world networks, the network is layered or clustered into LAN, MAN or WAN, accordingly, the agents are belong to different owner in different application domain with different goals and usages. In the complex scenario, it is not reasonable and not appropriate to propagate the reputation unchangeable. The reputation could be enhanced or decayed according to the trustworthiness between transmitting agents or the agent network structures.

5.3 Propagation mechanism design

Through the description in section 3, it is clear that designing ways of propagation is a critical task for spreading recognized reputation. The effectiveness of a propagation method can be analyzed from three perspectives: coverage, performance and structure. Wide coverage means that the reputation message should be propagated to as many agents as possible and as far distance as possible. High performance means that the propagation should efficiently make use of the network; the reputation message should reach the agents quickly without any distortion. Simple structure means the propagation mechanism should easy to implement and have a clear layered structure. According to those three principles, the thesis designs three forms of reputation propagation mechanisms: point-to-point based inquiry, broadcasting based propagation and observer based propagation. The next three subsections describe how those three mechanisms are designed, their detailed propagating processes, and their advantages and disadvantages.

5.3.1 Point-to-point based inquiry

The initial of idea of point-to-point based inquiry comes from the p2p system(Zhou Wen-li, 2006) which shares files among various nodes. Point-to-point based Inquiry is the simplest and most effective way of the reputation propagation. In order to have some knowledge of the target agent, an agent can send a request to its own acquaint agents. In case one of these agents also has interacting experience with the target agent, this acquaint agent may respond the request and send a reputation message back.

Figure 5.2 depicts the point-to-point based inquiry. The process of such inquiry can be summarized as the follow:

1. The initiative agent intends to start an interaction with the target agent.
2. The initiative agent sends reputation recommending requests to its neighbor agents according to its acquaintance list.
3. Upon receiving the request, the acquainted agent checks its interacting history list to see whether it has the past experience with the target agent. If it has, it will generate a reputation recommending message to the initiative agent. On the contrary, it will send a not-found message back to the initiative agent.

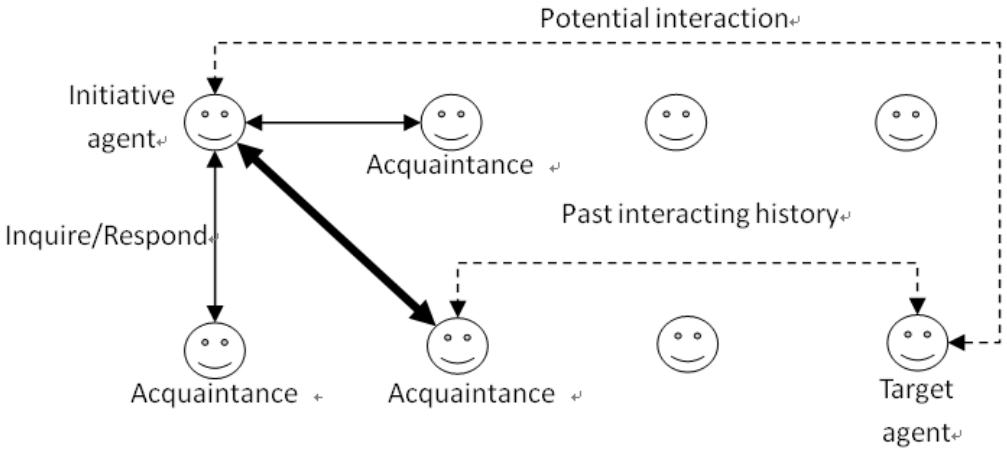

Figure 5.2: Point-to-Point inquiry - The process of the point-to-point based inquiry

4. The initiative agent receives responses from the acquainted agents. It filters the not-found message and combines the reputation recommending message based on reputation recognition algorithm which will be discussed in the later sections. Then the initiative agent can generate its own trust pattern against the target agent.

5. Depending on the judgment of trustworthiness toward the target agent, the initiative agent decides to or not to interact with the target agent.

The obvious advantage of the point-to-point based inquiry is its simplicity. Agents only need an acquaintance list to store their acquainted agents. The process of propagation is also a simple send and receives return. Such inquiry is actually the very basic form of reputation propagation. Other more complex forms like broadcasting and observer are derived from this basic form with a few add-ons to the way of propagation. The other advantage of the point-to-point based inquiry is its good performance. The initiative agent will not arbitrarily send too many requests out, they only send according to their acquaintance list. Such list can be limited to a relatively small size to avoid heavy network traffic. The disadvantage of the point-to-point based inquiry is that it is highly possible that all the acquainted agents in the list may not have the direct interacting experience with the target agent due to the size limitation of the list. Especially in a large computational agent society with a small acquaintance list, this situation would happen frequently. Such defect will counteract the above two advantages.

A slightly more sophisticated version of the point-to-point based inquiry is designed to allow the acquainted agents to exploit their acquainted agents again and again, i.e., constructing a chain of 2-layer or even n-layer propagation. Figure 5.3 illustrate such a reputation reference chain. Each agent first checks its own interaction history to see whether it has the previous interacting pattern with the target agent. If it has not, it will not stop at this point. It will try to forward this request to its own acquainted agents. Such process will go on until a reputation recommending message is returned form some nodes along the reference chain. This chained propagation solves the problem of limited acquaintance list for single agent. It composes a reputation network that allows agents search and propagate reputation. It enlarges the range of reputation inquiry for a single agent whereas keeps the acquaintance list reasonably small. But it is not to say that the depth of chain is deeper and better. Too long chain usually means heavy network transportation and worse performance. There must be some trade-off when choosing the depth of the reputation chain. Sacrificing some reputation propagation ability is to achieve better performance.

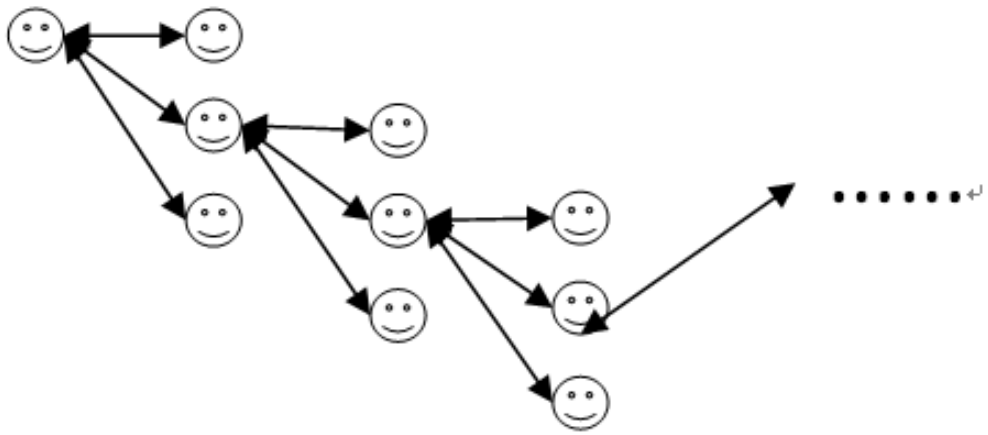

Figure 5.3: Chained point-to-point inquiry - The process of the chained point-to-point reputation inquiry

5.3.2 Broadcasting based propagation

The initial of idea of broadcasting based propagation comes from (Sandra M. Hedetniemi, 2006). They studied how the gossips propagated in the society in the form

of broadcasting. Compare to the point-to-point reputation inquiry, broadcasting can cover wider propagation scope in a short time. Once the initiative agent has interacted with some specific target agent, it will generate a reputation reference and send this reference to all the agents in its domain or society. Such broadcast is indiscrimination whereas the precondition is the initiative agent knows the identity and address of its neighbors.

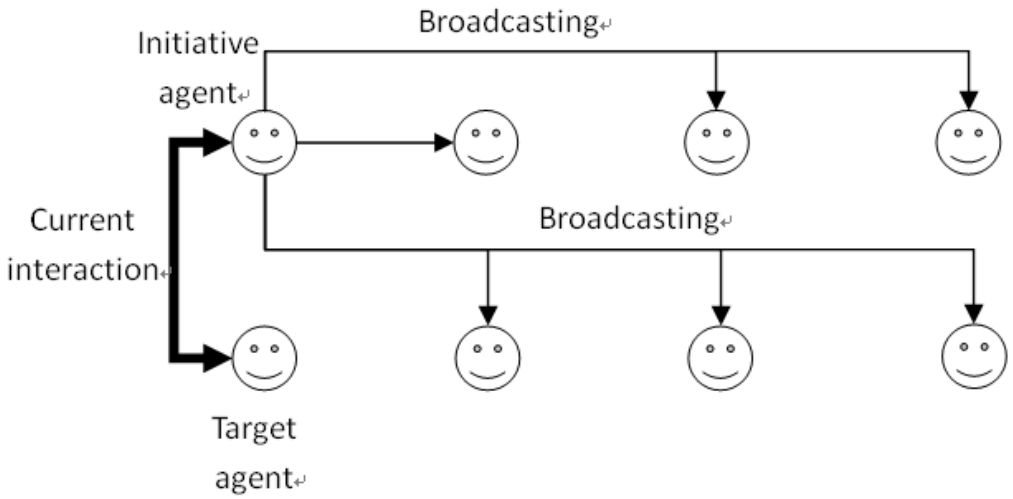

Figure 5.4: Broadcasting reputation - The process of broadcasting the reputation

Figure 5.4 illustrates how an agent broadcasts its experience to the other agents. The process can be summarized as the follows:

1. The initiative agent interacts with the target agent and the initiative agent updates its trust pattern through NTR algorithm mentioned in the previous chapter.

2. The initiative agent acquires the full agent list from the domain manager program and sends the perceived reputation reference out to all of the agents.

3. The receiving agent can choose to accept the broadcasting reference or simply just ignore the reference according to the strength of the relationship between the initiative agent and the receiving agent.

The biggest advantage of broadcasting is its wide coverage. Each agent tries to propagate its perceived reputation reference to the other agents. The reputation network is

updated frequently after each time of agents' interaction. This is very good for situation which needs closely watch the status of agent transaction or interaction. However, the biggest advantage can become the biggest disadvantage when used inappropriately. Broadcasting leads to performance problem because each time the initiative agent sends reputation reference to all the agents in its network. Such reference message will occupy too much of the bandwidth and sometimes it may heavily interfere the normal communication of the network and even block the whole network. Worse still, the broadcasting agent simply ignores whether the receiving agent has the requirement of the reputation reference or not. It sends the reference blindly and forces the receiving agent to accept it. In fact, there are only a few of the receiving agents who are interested in the reputation derived from the interaction between the initiative agent and target agent. This means the efficiency of broadcasting is less than the point-to-point based inquiry.

Size of the agent organization or society is the key factor that decides whether broadcasting is good or bad. The author's experiment shows, when agents built by JADE are deployed in a lab with 10 personal computers (2.4GHz, dual-core, 1Gb memory), 100M Ethernet connection, the turn point of using broadcasting based reputation propagation is 54 agents, that is, when the number of agents exceed 54, the broadcasting will severely impact the performance of the network. According to this discovery, the broadcasting propagation is not appropriate for system with vast number of agents. It is best suit for close agent society with very a few agents and with needs to closely watch the status of the agents' interaction.

5.3.3 Observer/Subscriber

In some cases, a better solution is to keep the wide coverage of the broadcasting while to avoid the uninterested agents being bothered to receive the reputation propagation from the initiative agents. In the objected-oriented software design, there is a design pattern called "Observer" (Hannemann, 2002), it is a pattern used to decouple the subject object and the observer object. In the paradigm of multi-agent system, agent is one kind of "intelligent object" so that it is wise to introduce this pattern into the world of multi-agent to build a solution that meets the needs aforementioned. The name can still follow the object-oriented way as the Observer, or more specifically, as

the Observer based reputation propagation. In this propagation method, agents can be categorized into four roles: subject, target, subscriber and unsubscriber.

The subject is an agent being watched by the other agent. Its behavior and its interaction with the other agents are interested by some other agents. The key to design a subject agent is that it holds a subscriber list that contains all the agents' identities who are interested in the subject. A target agent is the agent that interacts with the subject agent. The result of their interaction will be propagated as reputation reference. A register agent is the agent who needs reputation reference form the subject agent. It will register itself into the subscriber list of the subject agent hence it can receive reputation propagation message from the subject when the subject interacts with the target agent. An unsubscriber is an agent who totally has no interest in this subject. So it will never receive any piece of information from the subject agent.

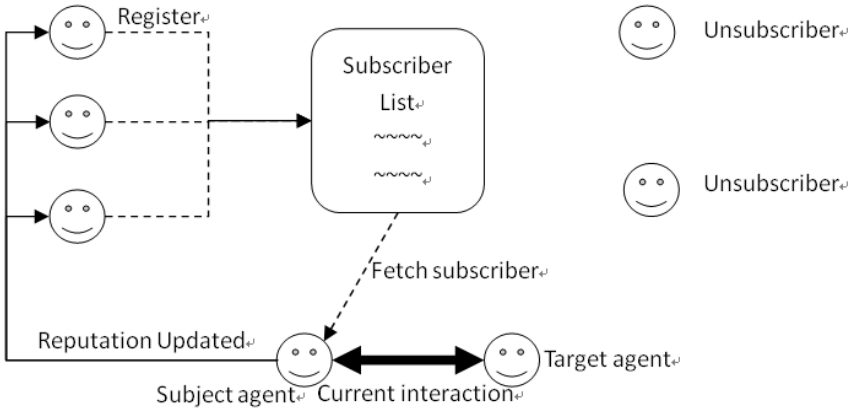

Figure 5.5: Observer - The process of observer based reputation propagation

Figure 5.5 illustrate how the subject agent notifies the subscriber agent its updated reputation. The process is summarized as the follows:

1. An agent is interested in a subject agent's behavior and it sends a registering request to the subject agent.

2. The subject agent receives this registering request and adds this identity of the request agent into its subscriber list.

3. The subject agent carries out an interaction with the target agent. The result of the interaction is transformed to update the trust pattern against the target

agent through the NTR algorithm mentioned in the previous chapter.

4. The subject agent checks its subscriber list and sends the perceived reputation recommending message to all the agents appeared in the list. The unsubscriber will not receive any message from the subject agent because they are not in the sending list.

5. The subscriber agent receives the reputation recommending message and updates its own trust pattern against target agent accordingly.

6. The subscriber is no longer interested in the subject agent. It sends an unsubscribe request to the subject agent.

7. The subject receives the unsubscribe request. It finds out the identity of the unsubscriber from its subscriber list and then it removes this identity. Next time when it carries out another interaction, it will not notify the removed unsubscriber.

The observer based reputation propagation combines the advantages from both the point-to-point inquiry based propagation and broadcasting based propagation. On the one hand, like point-to-point inquiry based propagation, the subject agent only serves those subscriber agents who register in advance. Other unrelated agents keep silent and the network traffic will not congest due to redundant communication. On the other hand, like the broadcast based propagation, the coverage of the observer based propagation is reasonably big. The subscriber list maintained by the subject agent is a controlled, ordered broadcasting list. Such a controlled broadcasting impact big enough zone of interested agents while save much time used to inquire or respond a single agent before every interaction. Thus the observer based reputation propagation is especially good for reputation propagation in Electronic commerce. In E-commerce, the users come from all over the world; it is infeasible to realize a global reputation network which allows individuals to freely exchange reputation. A better way to design the network is to use observer based propagation. Let some highly active users/agents become subject and the other user could choose to observe those subject user and gain reputation about specific dealer or product through the active users.

5.4 Reputation recognition

5.4.1 Reputation generation

In Chapter 6, computational agent can use the various transaction criteria as the input data to the STL algorithm. The algorithm then creates a matrix based trust pattern after repeated learning on these data. This matrix stores the experience gained through thousands of interaction, it is a "brain" designed for the computational agent to memorize the good behavior and the bad behavior. Chapter 6 points out that the value of the matrix based trust pattern is to learn the past interaction and to predict the possibility of success for the potential transactions. However, the matrix is only used in the sense of individual agent. For a multi-agent system, a learned trust do help agent improve its successful rate of transaction. But the complete advantages of multi-agent is not fully inspired unless the learned trust is put into the context of agent society, that is, generating reputation from the learned trust pattern and propagate those reputation to some interested agents. The algorithm for generating reputation is listed in the following pseudo code:

```
1. Send reputation request to reference agent;
2. Reference agent search patterns about target agent;
3. If (found)
    { target pattern vs. macro pattern;
    Got recommendation;}
4. Else { end generation;}
5. Return recommendation to host agent;
```

5.4.1.1 Drop point comparison

Initially, the basic trust pattern for an agent is a matrix with two types of drop points. The first type is success point, depicted as point in red, which represents successful transactions or interactions between the agent and its opponents. The size of the success point is proportional to the times of successful transactions or interactions. The second type is failure point, depicted as point in gray, which represents failed transactions or interactions between the agent and its opponents. The size of the failed point is proportional to the times of failed transactions or interactions. Figure 5.6 illustrates such a general pattern. This pattern is general matrix which is an assembled

graph that accumulates the drop points of many interactions with many agents. It does not aim to reveal the trustworthiness of a specific single agent. But it does show a macro pattern about what is a good transaction and what is a bad transaction for an agent if the agent is provided with transaction related data to learn or analyze. It provides a foundation about how the agent believes the trustworthiness is. To gain the reputation against one specific agent, the interaction history with this agent is still needed.

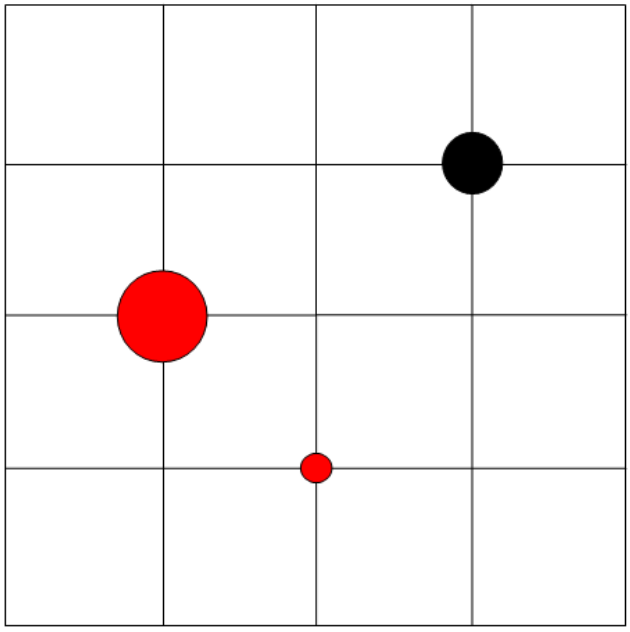

Figure 5.6: Macro pattern - The example of the macro trust pattern

The history of past interaction can also be recorded using the same style of pattern like the general trust pattern. The only difference is that the general pattern is design to contain all the drop points in all previous interactions. While the specific trust pattern is design to record the interaction between the initiative agent and the specified agent in the form of matrix and drop points. Figure 7 illustrates such a pattern. In this example, two features should be noticed. First, the drop points, both the success points and the failure points, are located in the same location as the general pattern. This is because the specified pattern is just a subset of the general pattern. The drop points appeared in the specified pattern must be accumulated to the points on the general pattern. Whereas the drop points appeared in the general pattern might not

appear in the specified pattern because some points may be plotted by some other interaction with some other agents. Reputation cannot be retrieved directly from the specified trust pattern as well because the specified pattern only shows the performance of one agent in limited interactions and there is no reference whether such performance deserves trust or not. A reasonable reputation should consider both the general pattern and the specified pattern so that the individual performance can be compared with the general performance.

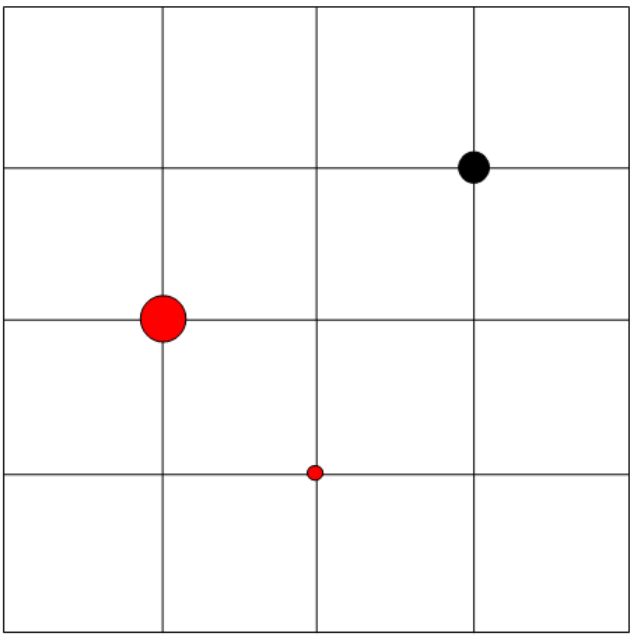

Figure 5.7: Agent pattern - The example of trust pattern for specified agent

The comparison between general pattern and specified pattern can be organized into a comparison on two folds. First, compare whether these two patterns have the same drop points. Usually, they will have. But in some cases, the specified pattern will miss some points that appear in the general pattern. Second, if some points lap over each other, it is necessary to compare the size of the points because the size indicates the strength of more trustworthy or less trustworthy. The size of drop points in the general pattern is definitely great than the size of drop points in the specified pattern. But how big the difference is can be an important indicator that how far the trustworthiness for specified agent biases the general pattern. This difference is also the source of the reputation. An agent receives a request to inquire a specified agent

about its reputation; it will check whether it has interaction history with the specified agent. If it does, it extracts the pattern for the specified agent and compares it with the general trust pattern. It will find some difference. If the difference is small (the drop points represent trustworthiness nearly overlap each other, the size is close), then the agent can send good reputation out to the inquiry agent. If the difference is big (the drop points represent untrustworthiness are overlapped or the drop points represent trustworthiness are missed, the size difference of the overlapped trustworthy points is big), then the agent can send bad reputation out to the inquiry agent.

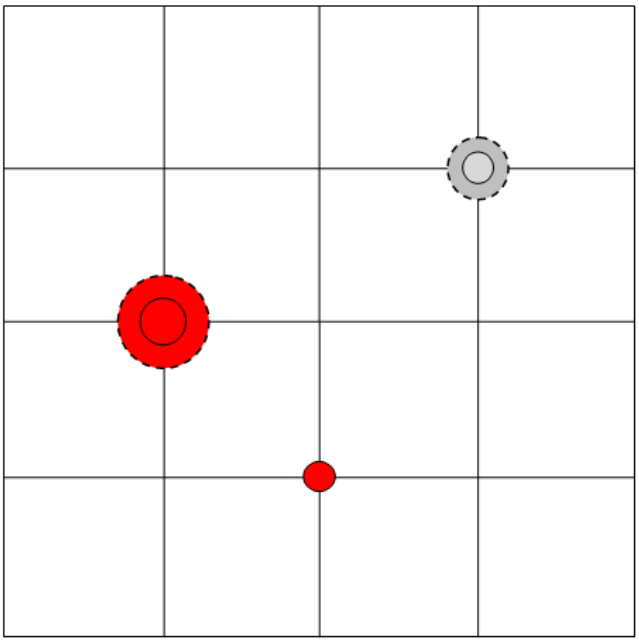

Figure 5.8: Result of comparison - The comparing result of the macro pattern and the agent pattern

5.4.1.2 Calculation of similarity

The calculation of the difference should be put into a mathematical framework so that the result of the calculation is a numerical value which can be propagated along the network, accepted and absorbed by the inquiry agent. Let P_g represents the matrix based trust pattern. Let d_{ij} represents the size of drop points on row i and column j; Let P_s represents the matrix based trust pattern. Let s_{ij} represents the size of drop

points on row i and column j. So the pattern is mathematically described as:

$$P_g = \begin{bmatrix} d_{11} & \cdots & d_{1j} \\ \cdots & \cdots & \cdots \\ d_{i1} & \cdots & d_{ij} \end{bmatrix}$$

$$P_s = \begin{bmatrix} s_{11} & \cdots & s_{1j} \\ \cdots & \cdots & \cdots \\ s_{i1} & \cdots & s_{ij} \end{bmatrix}$$

Let k, l represent the coordinate of the drop points in the matrix. Let T represents the total number of drop points in the general trust pattern. Let r_t represents the generated reputation value of the target agent. Let s_{kl} represents the successful drop points in the specified trust pattern. Let f_{kl} represents the failed drop points in the specified trust pattern. So the r_t is generated according to the following formula:

$$r_t = \frac{\sum s_{kl} - \sum f_{kl}}{T} \qquad (k = 1 \ldots i, l = 1 \ldots j, r \leq 1)$$

The first part of the formula calculates the rate of the successful drop points in the total number of interactions. The second part of the formula calculates the rate of the failed drop points in the total number of interactions. The value of r is actually reflects the difference between successful rate and failed rate, that is, a reputation value against specific agent which can be propagated to the other agents.

5.4.2 Received reputation

The initiative agent sends the reputation value to the inquiry agent. As section 5.1 shows, the value expresses the opinion about the target agent from the view point of the initiative (sender) agent. To simply accept this value as the reputation of the target agent is too rough. It is necessary to consider the sender's reputation as well. If a sender agent responds a reputation message about the target agent, while this sender's own reputation is awful. To believe such sender's recommending without any deliberation is dangerous. So the reputation receiving agent should also take the sender's reputation into account. This time, the generation of reputation is not by aid of the other agents. The process of generating reputation is controlled by the agent itself. The agent can guarantee that it will not be impacted by the biased influence at least.

Let r_a represents the accepted reputation, r_t represents the target reputation as stated in last section, and r_s represents the sender's reputation. So the r_a can be produced as the follow:

$$r_a = r_t(1 + r_s)$$

Depends on the value of r_s is positive or negative, r_t is enlarged or reduced. There are three situations:

1. $r_s > 0$, the reputation for the sender is positive, so the reputation of the target agent is also enlarged.

2. $r_s = 0$, an agent does not have the interaction experience with the sender agent. This absence of experience leads to set the sender's reputation as 0. This means if the agent does not have interaction experience with the sender agent, it will simply accept any reputation reference from the anonymous senders.

3. $r_s < 0$, the reputation of the sender agent is negative. The agent might have some unhappy experience with the sender. The reputation forwarded from the sender is discounted, the more negative the sender's reputation, and the more weakened the target agent's reputation.

5.4.3 Multi-source and decaying reputation

According to section 4.1, propagating the reputation through inquiry method may form a chain that connects multiple agents which forward or respond to the inquiry request. Whether the reputation propagated along the chain is decaying or lossless is an interesting problem. If it is lossless, the remote agents can send the intact reputation to the inquiry agent across many middle agents. Such situation is not good for huge agent society which is made up of many clusters of agents. An agent might be significantly influenced by a totally unrelated stranger from a remote cluster. On the contrary, if the reputation is decaying with a long distance between sender agent and receiving agent, the reputation might reach 0 before it can arrive at the receiving agent. Both the above two situations are not perfect choice. A good solution is to introduce a variable called "decaying coefficient", which can help adjust the degree of decaying speed. Then the designer or programmer of agents can control how far the reputation should propagate. For large agent society, the coefficient could be moderate so that

the distance of propagation is confined to reasonable clusters; for small agent society, the coefficient could be adjusted so that the reputation can be ensured to reach the farthest agent at the border of the society.

The variable is defined as , then the variable is added to the formula mentioned in the above section:

$$r_a = \gamma r_t(1 + r_s)$$

Without considering the situation of magnifying reputation (gossip), it is convenient to define the γ as $0 \leq \gamma \leq 1$. Let n represents the number between the sender agent and the receiving agent, N represents the total number of agents in an agent society. Then n can be used to decide the value of γ by using a sigmoid function:

$$\gamma = \frac{1}{1 - e^{-n}}$$

By substituting γ, the formula becomes:

$$r_a = \frac{1}{1 - e^{-n}} r_t(1 + r_s) \qquad (0 \leq n < N)$$

When an agent accepts the reputation, it needs to update its trust pattern against specific agent using this reputation. The updating method is to simply put the reputation as the input data for the NTR pattern. Just like the other input data, the reputation is another dimension which will contribute to the formation of the trust pattern. The process of how to emerge a trust pattern based on the input data is covered in Chapter 5.

5.5 Summary

This chapter mainly discusses the reputation related topics in the neural trust learning model. The definition of the reputation is first analyzed from five aspects. The reputation mechanism is then designed from three levels: reputation generation, reputation propagation and reputation receiving. Three propagation mechanisms are proposed to construct a reputation network. They are point-to-point inquiry, broadcasting and observer. The processes of each type are described in details. Then an approach of generating reputation based on the STL algorithm is designed so that the agent can produce reputation for other agents through investigating its own experience with the target agent. A decaying covariance is also introduced to deal with the problem of

reputation decaying in case the reputation is variable along the propagation chain. In the last section, the chapter proposes the approach of absorbing the reputation received from other agents, which is also based on the STL algorithm.

6

Results and Discussions

6.1 Introduction

The previous three chapters present a complete systematic neural trust model with trust learning algorithms, trust estimation algorithm and reputation mechanism. The focus is to describe the detailed design of the model and explain the rationales behind the model design. The focus of this chapter is to evaluate the proposed neural trust model from different aspects and analyze the results of the evaluations.

Before actually starting testing, an agent test bed is the necessity of the evaluation and should be carefully designed and implemented. It should follow some guidelines and choose an appropriate mature platform. In this chapter, JADE is the researcher's choice. The architecture of the test bed is presented and some of the implementation details are also presented in the chapter. The experiment environment, the experiment data preparation and the simulated testing data are demonstrated as well.

In order to evaluate the model, it is necessary to compare the model with other existing models. As analyzed in background chapter, most models are based on numeric or statistical foundation. The compared model should be built upon such mechanism, also, it should be easy to implement. The thesis take a simple numeric model to be compared with the proposed model. The comparison is carried out from two perspectives. The first perspective is to compare the quality of interaction through checking the successful rate of transactions, the second perspective is to compare the efficiency of discovering malevolent agents from all agents. The comparison is prepared as two stages. In stage 1, the simple numeric model is taken as the baseline to be compared

with the proposed model without the reputation propagation mechanism. The result of this stage is to exhibit the evidence that the learning capability of the proposed model is better than the simple numeric model. In stage 2, the proposed model without the reputation propagation mechanism is taken as the baseline to be compared with the proposed model with the reputation propagation mechanism. The result of this stage is to exhibit the positive influence of propagating reputation among agents. It also can reveal the fact that the socialized agents are better than individual agent on recognizing trustworthiness.

The chapter is made up of five sections. Section 2 describes the test-bed design and implementation, shows the requirements, architecture and components. Section 3 introduces the experimental preparations including environment configuration and data preparation. Section 4 presents the results through two comparisons. Section 5 discusses the results and explains the rationale behind the results. Section 6 summarize the chapter.

6.2 Test-bed design and implementation

For evaluating the effectiveness of the proposed computational trust and reputation models, a test bed is designed and implemented to provide an experimental platform. The test bed is built in Java programming language and it makes use of the agent framework JADE (Bellifemine et al., 2007) to provide agent management, agent communication and multi-agent environment. The test bed is based on the JADE API and JADE Agent Containers. The thesis chooses the experimental scenario as the electronic commerce activities that are finished by autonomous agents.

6.2.1 Objective and requirements

The objective of the test-bed is to build an automated experimental platform to test the feasibility and effectiveness of the proposed computational trust and reputation models. In order to realize the objective, the test bed needs to meet four basic requirements during its design and implementation.

- Agent oriented: The test bed should be designed and implemented as an autonomous agent based multi-agent system. The thesis aims to build a computational trust and reputation models for multi-agent systems. Thus the test bed

should be a multi-agent system. The architecture of the test bed should also be established from units of autonomous agent. This means the basic elements of the design are autonomous agents. The system is designed and functioned through modeling the interaction among different roles of agents. The techniques of object oriented programming are also used in the implementation level instead of the system design level (The programs create agents are also object oriented programs).

- Supporting multiple mechanisms: The focus of the research is the computational trust model and reputation mechanism. In order to state the superiority of the proposed models and mechanism, the test bed should not only support the thesis proposing mechanisms but also support the other popular trust and reputation related mechanisms. Furthermore, agents in the test bed should have the capability of choosing the different trust and reputation mechanisms as needed. Only with such capabilities, is it possible to let the agents in the test bed to be able to run in different trust/reputation mechanisms. So the researcher can compare the performance of different trust/reputation mechanisms and figure out the advantages and disadvantages of the proposed mechanisms.

- Simulating the real world e-commerce transactions: The autonomous agents in the test bed have to be assigned with some tasks so that they can interact for achieving the tasks. With the tasks, the test bed can reveal the results of the trust/reputation mechanisms. The tasks are organized in the form of a scenario. In this thesis, the researcher chooses the most ordinary electronic commerce as the scenario of the test bed. And the transactions take place in the e-commerce environments become the task of the autonomous agents. The test bed needs to simulate a virtual market with hundreds of autonomous agents which represent the authorizations and goals of the human users. The agents in the market will transact with each other to fulfill their bestowed goals. They will turn to exploit the trust/reputation mechanism before and after they start e-commerce transactions.

- Following standards: The test bed should follow the standard or popular techniques of multi-agent systems, that is, the test bed should be built upon the

existing agent framework. In such open source framework, many agent related functionalities are already in the state of maturity. It is convenient to directly use the application interface from the open framework instead of creating everything from scratch. The second level meaning of following standards is to let the design follow the standards in the field of agent programming. The current agreed industry standards for computer agents are Foundation International Physical Agent (FIPA) (FIPA, n.d.). The test bed should also follow this industry standard and avoid violation.

FIPA (Foundation for Intelligent, Physical Agents) specification was initially set up as a five-year mandate to specify selected aspects of MAS. It mainly focuses on the communication and interoperation specifications instead of the internal implementations of agents. The specification specifies the agent communication language (ACL) which includes content language, communicative acts and interaction protocols; it specifies the core communication support which includes naming, transport and directory service (it is also called Abstract Architecture); it also specifies additional communication support like agent management (mobility support, ontology service, and configuration management etc is still under drafting).

6.2.2 Development tools

To implement the test bed, the selection of the programming language is important. Theoretically, there is no compulsion in choosing the languages. Every language can be programmed to support a multi-agent system. But in the real world, an agent is a mobile program (mobile agent) that can cross hardware boundaries. The range of the information transmission and agent migration can be as large as the range of the Internet. The actual solution has to consider the heterogeneity of the computers, networks and software implementations. Java is the best language to tackle the difference because it is a programming language that is "write once, run anywhere". Its platform independence makes it the best choice for implementing a multi-agent system which allows a fully distributed agent deployment. The low level difference is hidden so that the researcher can focus on the design of interaction schemes and agent roles.

The widespread of MAS paradigm leads to the requirement of building good MAS based development tools and platforms. Both the academic communities and industrial

groups develop their platforms in the process of building the MAS projects (Gehao et al., 2009). In these projects, JADE (Bellifemine et al., 2007), FIPA-OS (Team, 2011), and ZEUS (Nwana et al., 1999) are the most prominent and prevalent platforms that support MAS development. The similarities among the three platforms are: all of them are based on Java programming language, all of them are open-source project, and all of them claim to strictly follow the FIPA (Foundation for Intelligent Physical Agents) specification.

In order to choose the best platform from the candidate platform, the researcher uses three tables to compare the them from different facets. Table 6.1 compares the compliancy of the three platforms taking seven layers of the FIPA 2000 specification as criteria. Y, N, P is inserted before the table: In this table, Y represents Yes, N represents No and P represents possible which means that the specific platform currently does not have the characteristic but leave interface to add such characteristic. Table 6.2 (Gehao et al., 2009) focuses on the maturity of the agent development platform through compare different platforms on monitoring and management, debugging, mobility, integration, and programming paradigm. Table 6.3 (Gehao et al., 2009) compares three platforms through their activity, popularity, accessibility and intellectual property form.

	Transport	**Encoding**	**Messaging**	**Ontology**	**Content Expressing**	**Communitive Act**	**Interaction Protocol**
JADE (Bellifemine et al., 2007)	Y	Y	Y	Y	Y	Y	Y
FIPA-OS (Team, 2011)	Y	P	P	Y	Y	Y	Y
ZEUS (Nwana et al., 1999)	P	P	Y	Y	N	Y	Y

Table 6.1: FIPA compliancy - FIPA compliancy for three platforms (Gehao et al., 2009).

	Monitoring & Management	**Debugging**	**Mobility**	**Integration**	**Programming paradigm**
JADE (Bellifemine et al., 2007)	Administration GUI	Event monitor and logger	LEAP	Web Service	API-based Programming
FIPA-OS (Team, 2011)	Wizard based configuration	DIAGNOSITCS class used to store debug message into file	Micro-fipaos	N/A	API-based Programming
ZEUS (Nwana et al., 1999)	Utility Agents and Visualizer	Visualizer check event happening	N/A	Wrapper	Visual Programming

Table 6.2: Platform maturity - Platform maturity for three platforms (Gehao et al., 2009).

Through observing the above three tables, the compliancy percentages for JADE, FIPA-OS and ZEUS are 100

Thus, JADE platform is a better choice for MAS development than FIPA-OS and ZEUS in FIPA compliancy, platform maturity and non-technical concerns. The test bed will be designed and implemented based on the JADE platform.

Java Agent DEvelopment Framework (JADE) (Bellifemine et al., 2007) is Java based agent platform developed by Telecom Italia Lab. It is an open source project and installation can be obtained free of charge from http://jade.tilab.com/ under GNU lesser GPL. It is fully FIPA compliant and on top of that it supports all FIPA message encodings (XML, Lisp, and binary). There is a direct support for SL, RDF, and XML content languages. It supports multiple containers that can be running on different machines forming one agent platform. FIPA compliant White and Yellow pages services are available. There is a support for an interconnected set of these services defined by the user. A subscription for these services is available too. Agent programming is based on Java language with additional support for behaviors and events. Both ontology and mobility are supported. There is no additional support for development of agents, but there is support for debugging which includes built-in message viewer and agent

	Activity	**Popularity**	**Accessibility**	**Intellectual Property Form**
JADE (Bellifemine et al., 2007)	Available and maintained	About 40000 downloads	From JADE homepage	Open source, general public license
FIPA-OS (Team, 2011)	The latest accounted activity is dated on 2003	About 50000 downloads	From Sourceforge.net	Open source, public domain, free of charge for non-commercial use
ZEUS (Nwana et al., 1999)	The last version is dated to 2001 and the ZEUS homepage was last updated in Jan 2001	No information provided	From ZEUS homepage	Open source, Mozilla Public License

Table 6.3: Non-technical concerns - Non-technical concerns for three platforms (Gehao et al., 2009).

introspector. Multiple network communication protocols are supported and additional ones can be developed via Message Transfer Protocol plug-ins. There is a very good support for different virtual machines like J2ME, J2EE, and also for .NET framework. The platform offers an scalability and communication speed. Security and web services are also supported. There is a collection of add-ons and 3rd party software available since there is a community of developers that are using JADE.

6.2.3 Test bed architecture

The test bed is made up of two major parts: The JADE platform and the agent activity zone. The JADE platform provides the regular management functionalities such as agent management, container management, and communication management. It creates an environment for agents communicating and interacting. It enforces the communicating protocol and the ways of agent behaviors. The agent activity zone is the workplace for agent transactions. Agents reside in their respective container and carry out transactions with other agents. According to the functions, the zone can be subdivided into three parts: autonomous agents, trust services and pattern libraries. The trust services and pattern libraries are called just like that they are the internal components of the autonomous agents. In order to explain their design mechanisms clearly, they are separated into independent parts. Figure 6.1 illustrates the architecture and components of the test bed.

The JADE platform is deployed as a typical distributed system. Every computer joins the platform becomes a "Container". Agents reside in these containers and the containers have the responsibilities to manage the agents. Among the containers, one of the computers plays a role called "Main Container". The main container might be a server with more powerful computing capabilities. It is the control center of the JADE platform. The main container manages a table called the container table which registers the object references and transport addresses of all the ordinary containers. Every container also manages a LADT table and a GADT table. The LADT represents the Local Agent Descriptor table and the GADT represents Global Agent Descriptor Table. The LADT manages a list for local agents. The container uses the LADT to find whether the message recipient lives in the container. The GADT caches the global agent information that the container receives, this help the system to avoid bottleneck of the main container. The AMS represent the Agent Management System. The AMS

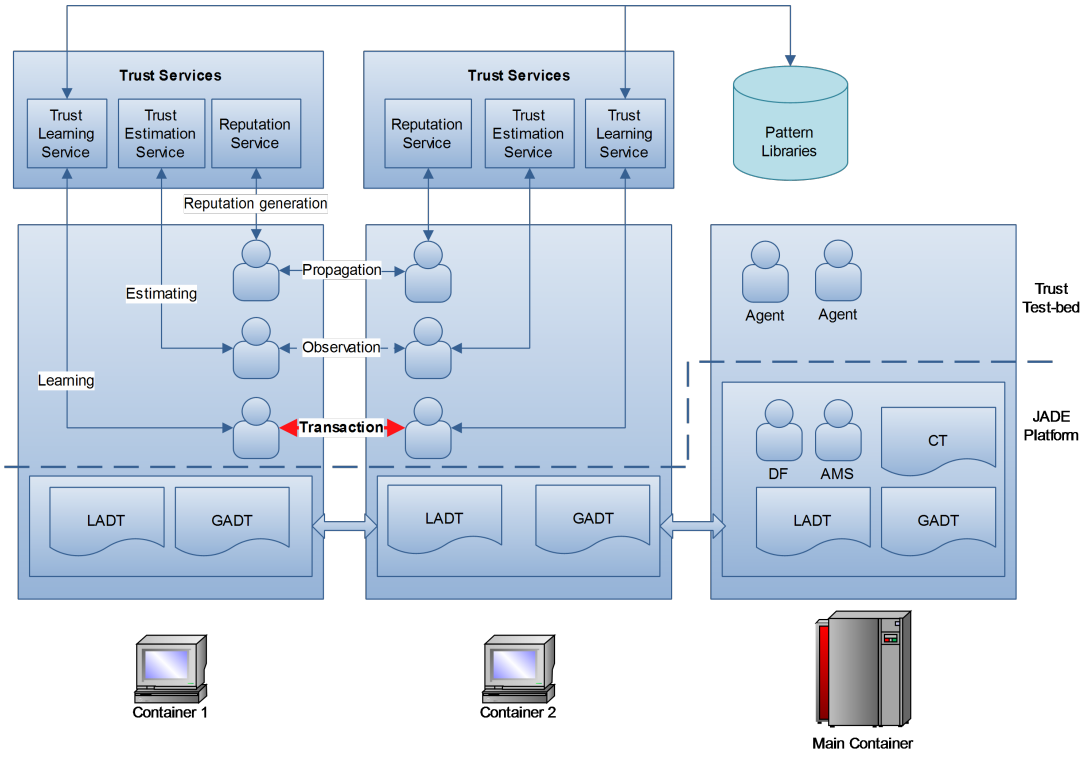

Figure 6.1: Test bed architecture - The test bed architecture for computational trust and reputation.

is an agent that monitors all the agents on the platform. It controls the life cycle of each agent and the agent that needs to start an interaction also first inquires the AMS for the contact information. Agents need to register to the AMS when they start up their lifecycles. DF represents Directory Facilitator and it is also an agent that is responsible for implementing the yellow page service. Agents can publish their service onto the yellow page so that the other agents can search for their preferred services. An agent can subscribe to the yellow page service. Once the service is updated or new services added, the agent will get notification for the changes.

Each agent has an agent identifier which helps the other agents search and interact with it. The identifier is a unique name globally. The name takes the form: $< local - name > @ < platform - name >$. The local name is the agent's name in the container and the platform name is the name of the container. Before starting an agent, it is necessary to start the JADE platform first (The platform behaves like an application server in client/server architectures). Once an agent is started, it sends registration request to the DF agent and it stays being alive until it is terminated by the AMS. Autonomous agents have goals and tasks. The task is represented as a behavior. There can be multiple behaviors assigned to one agent. JADE provides three types of behavior: one shot behavior, cyclic behavior and generic behavior. An agent can select one type of behavior or it can embed different types of behaviors to compose more complex compound behaviors. The communication methods among agents follow the FIPA specification and Agent Communication Language (FIPA, n.d.). There are a series of ACL primitives that covers the common terms required in communication. Fox example, INFORM represents sending a piece of information about some facts. CFP represents call for proposal from another agent. The messages sent between agents are asynchronous. Each agent maintains a queue to manage its receiving messages. The DF agent is started with the JADE platform and it manages the yellow page service for all agents. Agents publish their services to the yellow page and some other agents may try to search the services they have interests from this page. In the scenario of electronic commerce, the seller agent post their product selling services to the page when they come created and the buyer agents tend to search products from this page.

The trust services are the core components of the test bed design. The services include the trust learning service, the trust estimation service and the reputation service. These three services are independent with each other and each of them can be executed

concurrently. Autonomous agents use one of the services according to the stage that they step into. The trust learning service encapsulates the STL algorithm for learning trust patterns. The estimation service encapsulates the trustworthiness estimation algorithm. The reputation service encapsulates the reputation generation algorithm and reputation propagation mechanism. Every service is designed as one kind of behavior of the agents. Because behavior is one critical part of an agent, the behavior wrapped services are depended on the agent. When an agent starts its life cycle in the system, the trust services component generate an instance of the services as the form of a behavior and assign this behavior to the specific agent. Thus an agent owns the trust related behaviors after it comes into being. The ultimate goal of the services is to help agents achieve transactions with better quality. The behavior wrapped services should be plugged before and after the transaction behaviors so that they can take effect and protect the enclosed transactions. Before the transactions, the estimation service based behavior is executed so that the agent can predict the potential trustworthiness of its opponents. After the transactions, the trust learning service based behavior is executed so that the agent can learn from the finished transactions and improve its cognition toward the specific agents.

Pattern libraries store and manage the patterns learned from previous transactions for agents. The pattern libraries are not central to all the containers and agents. In the deployment, the solution is to assign one pattern library only for one agent. The life cycle of an agent may be temporally terminated but the pattern library is kept in the database forever. Such design is to ensure the experiences learned from the history won't be lost due to the interrupt of agents' life cycle. When an agent is revived by the AMS, it retrieves its patterns from its own pattern library to recreate its belief of trust. The pattern library is simply made up of three tables: TransactionPatterns table, AgentPatterns table and MacroPatterns table. They are corresponding to the three types of patterns mentioned in Chapter 5. The relationships among them are an accumulating relationship. The TransactionPatterns table is the basic one that collects all the previous data about transactions. It records the pattern for each single transaction done by the agent. The AgentPatterns table is a view by accumulating the patterns of a specific agent in the TransactionPatterns table. The MacroPatterns table stores the most general patterns which are accumulated by all transactions and all agents. Patterns are the most important data stored in the library. It is a matrix like

data so the researcher uses a generic object to store the pattern data in the program. It appears as binary data in the database tables. The marking attribute illustrate the results of the transactions. It is a Boolean value because the transaction is either successful (true) or failed (false). The MacroPatterns table does not need the marking attribute because the patterns recorded in the table combines all the successful patterns and failed patterns to create a macro pattern which contains the emerged feature for both successful transaction or failed transaction. The library is implemented as a cluster of databases. The physical implementation is based on MySQL relational database. Figure 6.2 illustrates the conceptual model of the pattern library.

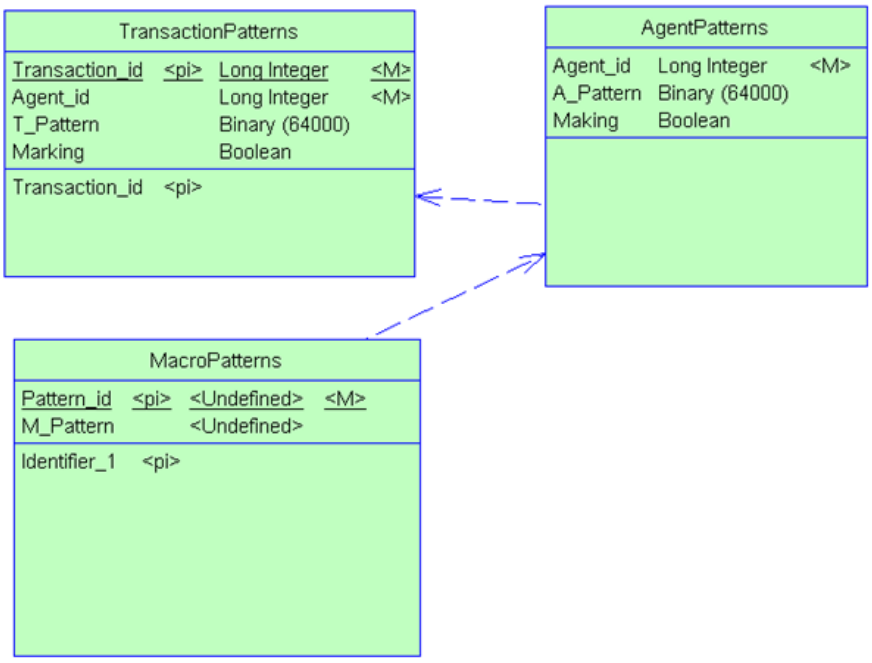

Figure 6.2: Pattern library - Conceptual model of the pattern libraries.

6.2.4 Test bed implementation

Figure 6.3 illustrates the package diagram of the test bed. The Sellers and Buyers package contains the seller agents and buyer agents. The behavior package contains the regular behavior classes for carrying out transactions. The additional trust related behaviors are accomplished through the Learning-Services package, Estimating-Service package and Reputation-Service package. The seller and buyer agents install the be-

haviors from the Behaviors package and finish the tasks in the way of the behaviors. The UI package contains the classes related to the graphical user interface. The process of transactions, the display of the learned patterns are all realized through this package. The VirutalMarket package contains the utility class and constant class that represents the configuration of the virtual market and it also contains the product classes that are to be sold in the multi-agent system.

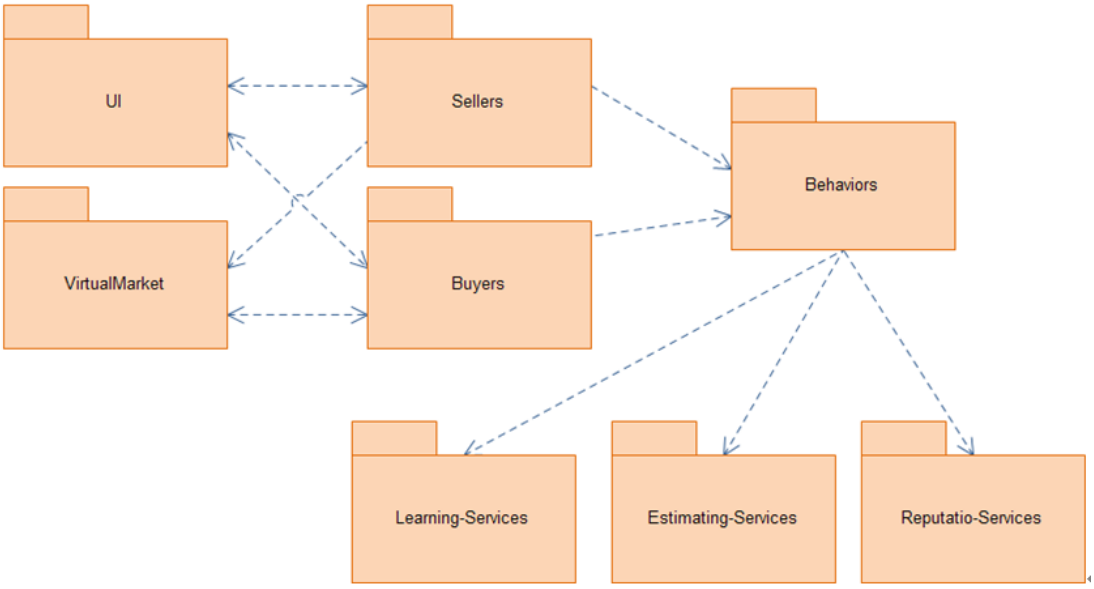

Figure 6.3: Test bed implementation - Test bed package diagram.

The key classes of the implementation are the buyer agent class and the seller agent class. The thesis takes the buyer agent as an example to illustrate the internal structures. Figure 6.4 is the implementation diagram for the buyer agent. The figure is drawn in Unified Modeling Language and the connection is the association relationship among classes. In this figure, the system organizes the functionalities around the BuyerAgent. The behaviors of the buyer agent can be divided into two categories. The first category is the common transaction related behaviors. For example, the TransacTicker class plays a role of controlling the time. If the waiting time is up, the agent stops sending CFP request to the other agents. The purchase behavior finishes the regular transactions through messages based communication. The other category is the trust related behavior. The thesis uses the Strategy pattern to create a flexible selection scheme of trust behaviors. The design pattern allows the researcher to switch the trust

related behaviors quickly and adds new algorithm based behaviors easily. According to section 2.1, the test bed can compare different trust mechanisms and help the researcher evaluate the proposed algorithms and mechanisms. The market model defines the configuration details of the context. For the buyer agent, the MarketModel class give information that facilitates the potential transactions.

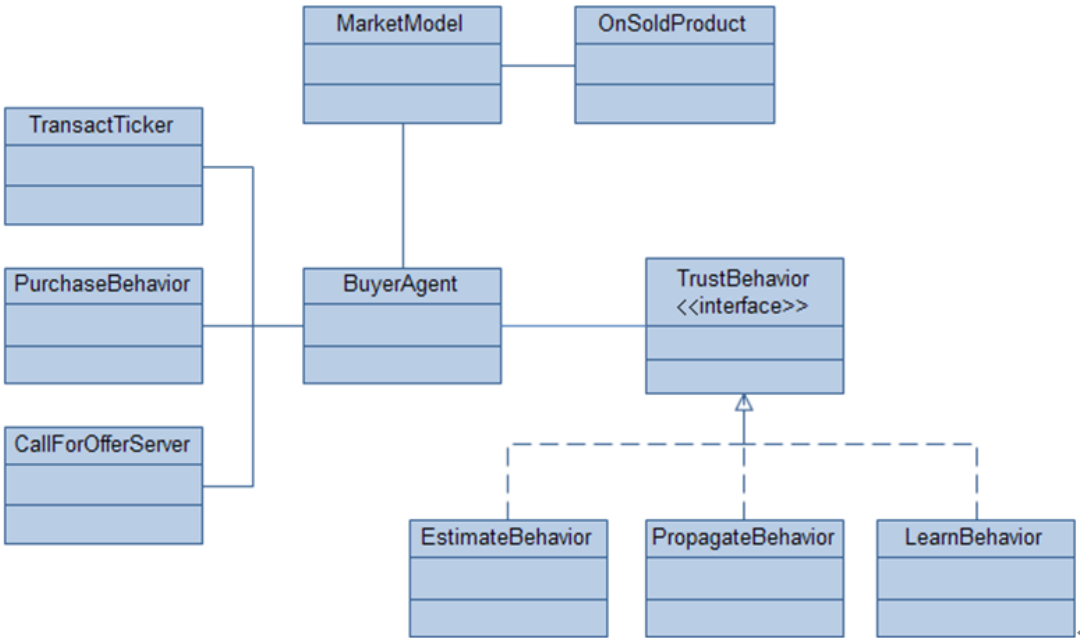

Figure 6.4: Implementation of the buyer agent - Class diagram for implementing the buyer agent.

6.3 Preparation

6.3.1 Experimental scenario

The experimental scenario for testing the computational model and reputation should be simple enough so that it can reveal the advantages and disadvantages of the specific algorithms and mechanisms. At the same time, the scenario should have realistic significance so that the researcher can have intuitive experiences about the pragmatic value of the study. The common applications for multi-agent systems include games, web assistants, computer aided planning, industry process controls, automated electronic

commerce, etc. The thesis choose the automated electronic commerce as the experimental scenario because its simplicity and understandability. It might be the most familiar type of interaction for the readers. The major interactions in the electronic commerce are transactions and the objects or domains for the transactions are simply the product being transacted. This scenario gives the researcher a clear structure to design the test bed and to test the algorithms.

The scenario is automated e-commerce. The word "automated" means that the transaction, including searching products, enquiry, negotiation, placing orders, payments, deliveries, are all done by the autonomous agents. At this stage, the technical development cannot avoid human intervention. Some critical authorizations and decisions have to be made by human users. The agents here behave like a representative of the transactions. The evaluations of the transactions have to be done by the users who authorize the agents to transact unless someday the agents' capability of cognition reaches the highness of understanding the goodness and badness of the transactions.

There are two roles of agents in the scenario: buyer agents and seller agents. The buyer agent is trying to search their preferred products from the service list published by the seller agents. The seller agents and buyer agents will communicate and transact with each other under the interaction protocol as mentions in previous section. The seller agents are more complicated. They can hold two types of attitudes during the transactions: benevolent attitude and malevolent attitude. The probabilities of providing good products from the benevolent seller agents are high, while the probabilities of providing good products from the malevolent seller agents are low. The judgments about good products are achieved through reviewing the dimensional data of the product domain. The major task of seller agent is to investigate how the computational trust and reputation can help buyer agents avoid inferior transactions with malevolent seller agents.

Books are the default product selling through the autonomous agents in the experimental scenario. Thus the domain of the product is the book. The dimensions for describing a book transaction are set as reputation, price, quality, delivery, after-sale. Reputation refers to the transferred reputation from the other agents, price refers the soundness of the price level, quality refers to the cognition of the product quality, delivery refers to the satisfaction of the delivery speed and quality, after-sale refers to the cognition of the after-sale service quality. For simplifying the evaluation of the

dimension, every dimension is marked as 1 for good and 0 for bad. Such simplified setting is also convenient for the learning algorithm to read the product dimensional data as input vectors. The simplification is only for easier understanding. In the real case, the dimensional input is provided by the human users and the value is between 0 and 1 according to their subjective judgment. When the user feel a dimension is not good enough, he or she will give a value close to 0. In the experiments, the input data is actually generated as decimal value between 0 and 1.

6.3.2 Data preparation

The experimental data are mainly the simulated transaction data that appears in the form of production dimensions. After every round of the transaction between two agents, one line (vector) of the data is provided to the buyer agent. There are two groups of transactional data. The first group is training transactional data. Data in this group are generated according the proportion of benevolent agents and malevolent agents. For each seller agent, there is a file assigned for it to be its simulated transactional data. The second group contains the test data (observation data). This group is design for testing the ability of differentiating bad transactions of the buyer agents after learning the computational trust through the first training group. The data of these two groups are generated through transactional data generating program. In the real world, all transactional data should be provided by human user who owns the agent and authorize the agent to represent him/her.

To simulate the transactional data of thousands of agents, it is not realistic to manually design the data from scratch. The better way is to set the rule of data generation and let the computer program to produce the data for testing. The researcher designs a simple program to do this job. The algorithm of data generation are described as follows:

```
1. While(!end of file){
2.     if(benevolent){
                        Marking = 1;
                        At least 3 random slots = 1;
}
3.     else if(malevolent){
                        Marking = 0;
```

```
                        At least 3 random slots = 0;
4.      next line;
}
```

In this data generation algorithm: A new file is created for one agent. The slots and marks in this file are all initialized to 0. The algorithm will change the value of the slot and the value of the mark according to the rule in the algorithm. The size of file is 100 lines. Each line is the data of five dimensions plus the marking. The marking represent the subjective evaluation of the transaction as a whole. All the value in the table is restricted to either 1 or 0. "1" is positive result and vice versa. The transactional data is either for benevolent seller agents or for malevolent seller agents. The thesis set that one half of the seller agents are benevolent and the others are malevolent. For the benevolent agent, the probability of being benevolent is 80%, it still have 20% chances to be malevolent. The method of generating benevolent data is: *Marking = 1, for the other 5 slots, assign 1 to more than 3 slots. The slot being assigned is randomly specified.* For the malevolent agent, the probability of being malevolent is 80%, where there is a 20% chances of being benevolent. The method of generating malevolent data is: *Marking = 0, for the other 5 slots, assign 0 to more than 3 slots. The slot being assigned is randomly specified.* Then generate the next line. If the line number reaches 100, then quit and start again for another new agent file. [1]

Table 6.4 is an example of simulated transactional data. Each line evaluates a transaction from 5 dimensions. The marking is a subjective evaluation to the transaction as a whole. The example is a fragment extracted from a benevolent agent's transactional data file.

Reputation	**Price**	**Quality**	**Delivery**	**After-Sale**	**Marking**
1	0	1	0	1	1
1	1	1	1	1	1
0	0	0	0	1	0
0	1	1	1	0	1

Table 6.4: Examples of simulated data - Examples of simulated transactional data.

[1]The thesis provides an installation guidance and testing report as the the attachments. The source code of the test bed can also be downloaded from http://www.ynsoft.org.

Table 6.5 gives a summary of the experimental data. The number of agents attended to the scenario is 100; and 50% of them are buyer agents and the other 50% are seller agents. The attitudes of the seller agents are also split evenly: 50% benevolent and 50% malevolent. That means there will 25 benevolent seller agents and 25 malevolent seller agents. The number of simulated data files is equal to the number of agents. The size of the data files actually equals the number of transactions that one agent will do during its life cycle. Thus the number of transactions are 150,000.

The Total number of Agents	100
Number of Buyer Agents (50%)	50
Number of Seller Agents (50%)	50
Benevolent Seller Agents	25
Malevolent Seller Agents	25
Number of transactions	150,000
Number of simulated data files	50
Size of the data file (lines)	60

Table 6.5: Summary of data preparation - Summary of data preparations is listed in this table.

Table 6.6 shows the configuration of the experiment environment. The test bed is deployed in a distributed manner. The main container is a Server with 2.13 GHz Xeon CPU and 4GB memory. The ordinary containers are three PCs with Intel Conroe i3 2100 CPU and 2GB memory. The operating system for the main container is Window 2003 Server and the operating system for the containers is Windows 7. All the computers are connected with 100M LAN.

Hardware	1 Server (IBM X3400 M3, Xeon 2.13GHz, 4GB memory) as the main container, 3 PCs (Lenovo i4160, Intel Conroe i3 2100) as the container. Each container runs 250 agents.
Software	JADE platform and Trust Test bed, Window Server 2003 for Server and Window 7 for PCs.
Network	100Mb LAN

Table 6.6: Summary of experiment environment - Summary of experiment environment and configuration.

The simulated data is generated through program and the initial marking of the transactional data is also allocated by the program according to the predefined benevolent/malevolent probability. This might be different from the real data which is generated from the real transaction and marked by the real human user. However, this won't negatively influence the validity of the test bed because the recognition of trust will reflect patterns according to the simulated data in case the data do have some regularities. In the real world, human input tends to be even less regular than the input generated from computer programs. So once the model works fine in the simulated data. It will reveal the pattern or regularity from the transactional data marked by the real human users.

6.4 Results

After building the experiment environment, it is time to use the test bed to evaluate the proposed framework and its internal algorithms and mechanisms. The most direct choice is to compare the proposed framework with some other trust mechanisms. The precondition of the comparison is that both trust frameworks should be working in the same platform and process the same data in the same duration. The results of the comparison mainly focuses on two aspects: operational effectiveness and operational performance. The operational effectiveness means that the specified algorithm or mechanism can effectively finish the designed jobs and achieve satisfied qualities. The operational performance means that the specified algorithm or mechanism can finish the designated jobs as timely as possible. Theses two aspects are also criteria for successfully evaluating the proposed model.

6.4.1 The compared algorithm: a simple number based model

There are many computational trust related mechanisms and algorithms developed in the past few years. Chapter 3 has introduced some typical algorithms in detail. Majority of these trust/reputation models are based on number or statistical methods. In this section, the research chooses a simple but popular model to be compared with the proposed framework. This model is based on a simple number based method: addition and subtraction. The trust is represented as a numeric value. The initial trust value is 0. After each successful transaction, the agent simply adds 1 to its trust

value. On the contrary, after each failed transaction, the agent simply minus 1 from its trust value. The sum of the addition indicates the trustworthiness of the agent toward another agent. The agent sets a threshold to be the judgment rule on decision making. For example, if the threshold is set as 30, an agent trust value less than 30 will not enter any transactions with the host agent. The trust value can also be negative number which means degree of trustworthiness is none and the untrustworthiness is big. Although such model is relatively simple, it is widely accepted as the basis of many successful e-commerce web sites. For example, the customer rating system of Taobao (Taobao, 2011), EBay (Ebay, 2012) are all advanced variations from the basic number based model. To use the number based model, the comparison can be close to the real world application. Additionally, the simple model is easy to implement on the test bed. This saves great effort in designing and programming.

6.4.2 Simple number based model vs. Neural trust model

The first experiment is to compare the successful rate of buyer agents between the proposed framework and the number based model. This is a test especially designed for evaluating the operational effectiveness of the algorithms. The successful rate refers to the rate of successful transactions among all the transactions. Such rate can reflect two things about the trust models. The first thing revealed is that the high-low of the ability of avoiding malevolent agents. The second thing revealed is the accuracy of finding benevolent agents. When facing thousands of seller agents, the inferior model tends to make many wrong decisions and transact with malevolent agent thus decrease the rate of successful transactions.

There are 100 agents participate into the experiment. One half of them play the role of buyer agent and the other half plays the role of seller agent. Among the 50 seller agents, 25 of them are malevolent. The experiment is designed to see which model can achieve better and higher successful rate. The simulated transaction data contains 150,000 lines of transactional data. The testing time is about 4 hours (15000 seconds). The sniffer programs are set in the test bed to record the number of successful transactions at the sampling time. The sampling interval is 1500 seconds. Both models are first trained with a set of prepared data listed in section 3.2.

Figure 6.5 illustrates the result of comparing the specified two models on the successful rate of transactions. The successful rate refers to the rate of transactions that

the result is marked as positive 1 by the end user. This rate is a measurement to reflect the degree how the trust model helps the agents improve their transactional quality. The horizontal axis is the time used in the transactions. The range of the time is from 1500 seconds to 15000 seconds. The vertical axis is the rate in percentage. The range is from 0% to 100%. The two lines in the figure represent the two types of models to be compared. Trust1 line represents the number based model. Trust2 represents the proposed model. Both lines start from 1500 seconds. It is clear that Trust2 line is superior to Trust1 line from the start point. The initial rate of success reaches 13% for Trust2; the Trust1 is only 1%. The difference is kept until the end of all transactions. The final rate of success for Trust2 reaches 98% while the final rate of success for Trust1 is only 36%.

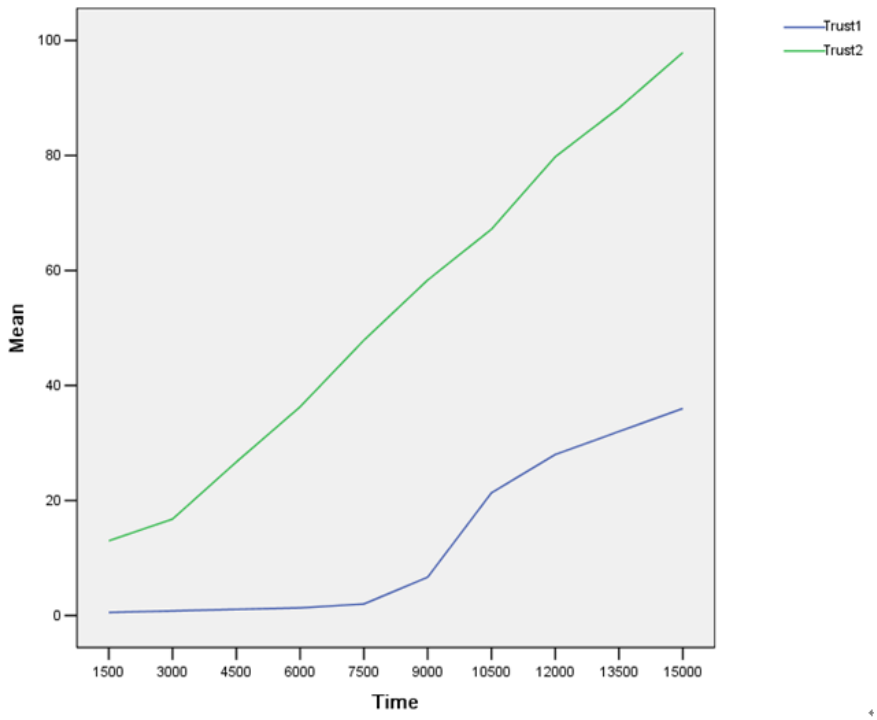

Figure 6.5: Simple number based model vs. Neural trust model - Successful rate for the compared models.

The second experiment is to compare the elimination speed of malevolent agents between the proposed framework and the number based model. This is a test especially designed for evaluating the operational performance of the algorithms. Elimination

refers to finding out the malevolent agents and let agents stop any further transactions with the malevolent ones. Just like the malevolent agents are eliminate from the electronic market. A better algorithm or a better model needs eliminate the malevolent agent as quick as possible so that the agents can avoid unnecessary loss in the future transactions.

There are 100 agents participate into the experiment. One half of them play the role of buyer agent and the other half plays the role of seller agent. Among the 50 seller agents, 25 of them are malevolent. The experiment is designed to see which algorithm can find all these malevolent agents as quick as possible. The simulated transaction data contains 150,000 lines of transactional data. The testing time is about 4 hours (15000 seconds). The sniffer programs are set in the test bed to record the number of eliminated agents at the sampling time. The sampling interval is 500 seconds. Both models are first trained with a set of prepared data listed in section 3.2.

Figure 6.6 illustrates the result of eliminating malevolent agents in a line diagram. In this figure, the horizontal axis indicate the time that the test bed executing. The range of time is from 500 seconds to 15000 seconds. The vertical axis is the mean number of eliminated agents for both models. The range of the number is from 0 to 25. There are two lines in the figure. The line named Trust1 represents the number based model and the line named Trust2 represents the proposed model. In the first 1000 seconds, both models are uneventful and do not find any malevolent agents. Within 1500 seconds, the proposed model starts to find the first malevolent agent and the Trust2 line exceed the Trust1 line. The difference of two lines keeps magnifying. The Trust2 reaches the top value of 25 at the time of 12500 seconds. It keeps this number until the end of the experiment. The Trust1 only reaches 11 at the time of 14000 seconds. This indicates that the number based model does not find all the malevolent agents.

6.4.3 Reputation vs. Non-reputation

The above experiments compare two different models to evaluate the operational effectiveness and performance effectiveness of the proposed framework. Next, the thesis turns to the test the reputation related behaviors. The goal of the next comparison is also changed to check the influence of the reputation. Thus the target of comparison is no longer the number based trust model. Now the comparison will be carried out

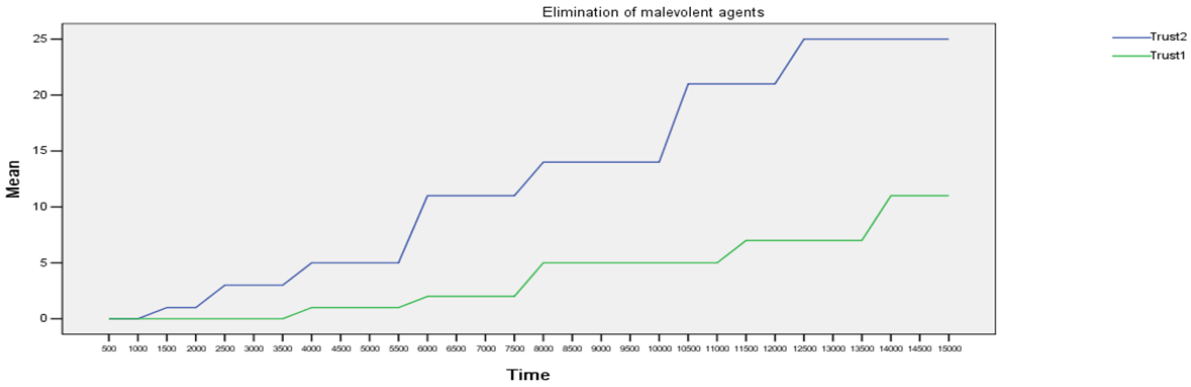

Figure 6.6: Simple number based model vs. Neural trust model - Result of eliminating malevolent agents.

between two instances of the proposed framework. One instance is only equipped with trust learning algorithm and trust estimation algorithm. The other instance is not only equipped with all the algorithms of the first instance but also it is enhanced with the reputation generating and propagating mechanism. The key to the comparison is to reveal the impact of adding reputation into the transactions.

The first experiment is designed to compare the successful rate of buyer agents between the complete framework and the framework without reputations. The results can also help evaluate the operational effectiveness of two configurations. The comparison needs to prove that the existence of the reputation mechanism can improve the effectiveness of the whole framework. The speed of finding malevolent agents should also be improved. Furthermore, this experiment is a chance to explain why multi-agent systems work better than single intelligent agent.

There are 100 agents participating into the experiment. One half of them play the role of buyer agent and the other half plays the role of seller agent. Among the 50 seller agents, 25 of them are malevolent. The experiment is designed to see which configuration can achieve better and higher successful rate. The simulated transaction data contains 150,000 lines of transactional data. The testing time is about 4 hours (15000 seconds). The sniffer programs are set in the test bed to record the number of successful transactions at the sampling time. The sampling interval is 1500 seconds. Both models are first trained with a set of prepared data listed in section 3.2.

Figure 6.7 illustrates the result of comparing the specified two configurations on the

successful rate of transactions. The horizontal axis is the time used in the transactions. The range of the time is from 1500 seconds to 15000 seconds. The vertical axis is the rate in percentage. The range is from 0% to 100%. The two lines in the figure represent the framework and the framework without reputation respectively. They are called WithRepu line and TrustOnly line. Both lines start from 1500 seconds. The WithRepu line is superior to TrustOnly line at the starting point. The initial rate of success reaches 25% for WithRepu line; the TrustOnly is only 13%. The difference is kept until the end of all transactions. The final rate of success for WithRepu reaches 98% while the final rate of success for Trust1 reaches 94%. Although the final rate is close enough, the WithRepu quickly approach the maximum success rate while the TrustOnly approaches relatively slowly.

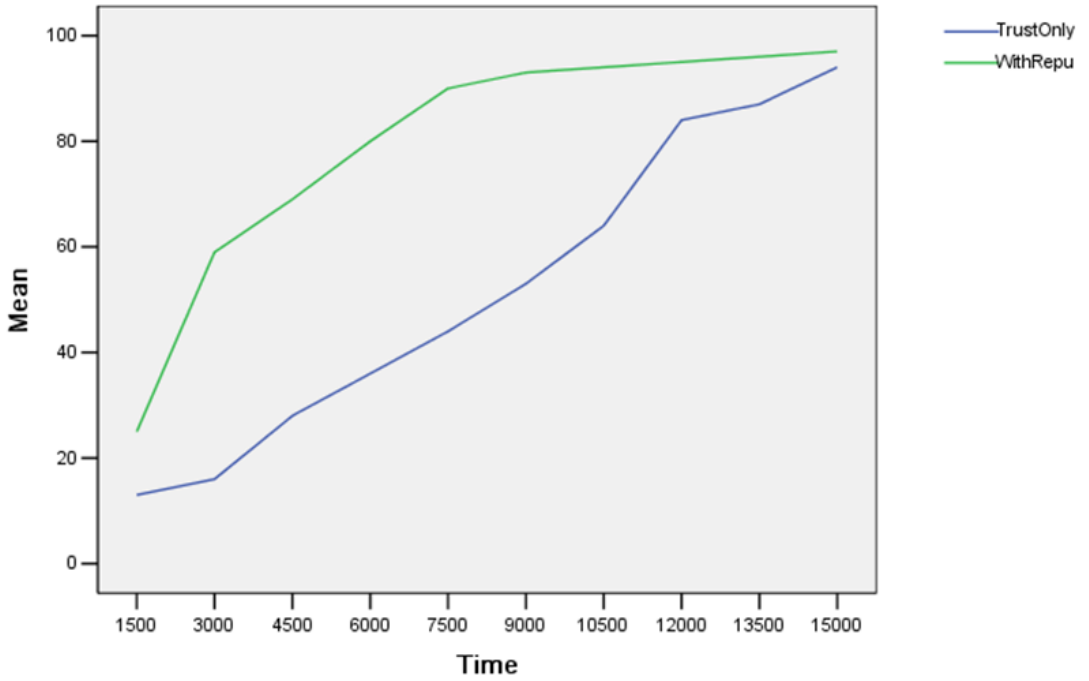

Figure 6.7: Reputation vs. Non-reputation - Result of comparing frameworks with and without reputation support.

The next experiment is to compare the elimination speed under the condition with reputations and without reputations. The existence of reputations may influence the operational performance of the framework. The common sense about reputations is that an agent can avoid many unnecessary transactions if it is informed by the other agents in

the form of reputations. Thus, the malevolent agents should be eliminated much more quick in an environment with reputations than in an environment without reputations. Additionally, the experiment can discover the degree of influences produced by the reputations.

There are 100 agents participating in the experiment. One half of them play the role of buyer agent and the other half plays the role of seller agent. Among the 50 seller agents, 25 of them are malevolent. The experiment is designed to see which configuration can find all these malevolent agents as quick as possible. The simulated transaction data contains 150,000 lines of transactional data. The testing time is about 4 hours (15000 seconds). The sniffer programs are set in the test bed to record the number of eliminated agents at the sampling time. The sampling interval is 500 seconds. Both configurations are first trained with a set of prepared data listed in section 3.2.

Figure 6.8 illustrates the result of eliminating malevolent agents in a line diagram. In this figure, the horizontal axis indicate the time that the test bed executing. The range of time is from 500 seconds to 15000 seconds. The vertical axis is the mean number of eliminated agents for both configurations. The range of the number is from 0 to 25. There are two lines in the figure. The line named TrustOnly represents the configuration without reputations and the line named WithRepu represents the configuration with reputations. Two lines separate from the start point at 500 seconds. The WithRepu line lead the number of more eliminated malevolent agents from the beginning. Its advantages are magnified with the elapsing time. It first finishes eliminating jobs at 7500 seconds. Comparing to the TrustOnly line which reaches the top at 14000, it precedes for 6500 seconds. This saves one half of the transactional time.

6.5 Discussion

6.5.1 Analysis of testing scenarios

In the comparison between the proposed framework and the number based model, the final successful rate of the former precedes the later 62% (98%-36%). This difference is significant. Both models are trained before they are fed with the testing data. Both of them have generated specific type of memories. For the proposed framework, the memory is patterns learned from the training data; for the number based model, the memory is the trust value accumulated in the previous transactions. During the testing

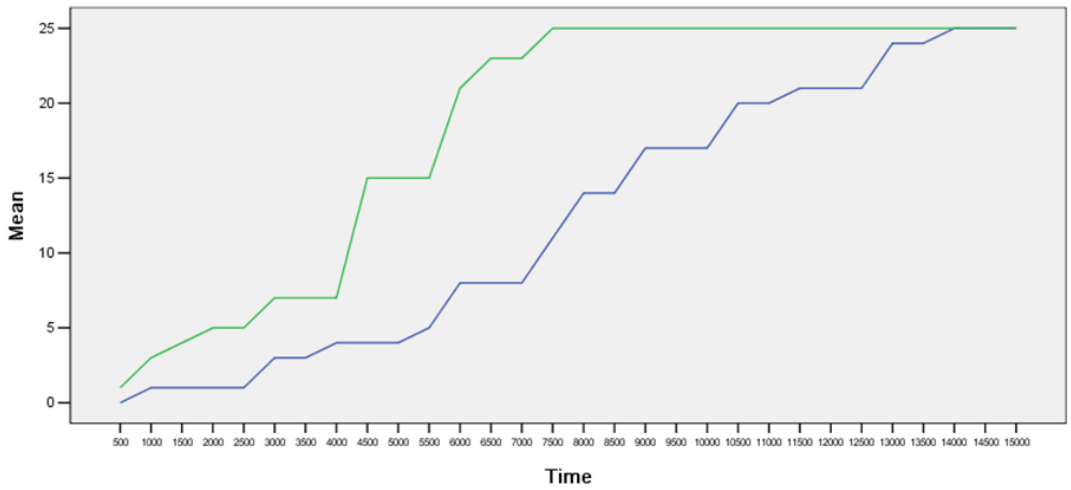

Figure 6.8: Reputation vs. Non-reputation - Result of eliminating agents between the proposed framework with and without reputations.

phase, the data files used for both models are new transactional data. In these files, agents are new agents that haven't interacted with the buyer agents under testing. The transactional data are also fresh to agents for both models. In this case, the macro patterns stored in the pattern library become a solid foundation for dealing with the new agent and new data. The macro patterns are general patterns that reveal the general features of the so called good behaviors. The trust estimation algorithm can recognize the new transactional data and generate a potential pattern. Then the pattern is compared with the macro pattern. That's why the framework can quickly adapt to the new testing environment. On the contrary, though the agents in the number based model have trust value to memorize the specific agent's behavior, the trust value only takes effect when the agent meets the same agent in another round of transaction. In the testing, nothing is familiar to the agents in the model. This leads to a very slow adaptation to the environment. The agent has to interact with unfamiliar agents first and learn from successes and failures from scratch. Figure 6.6 shows that during the first 9000 seconds, the successful rate is really low. After 9000 seconds, the successful rate has significant increasing. The number based model has to use lots of time and lots of negative transactions to be adapted. This is an unacceptable cost that should be avoided.

The speed of eliminating malevolent agents for the proposed framework is also

higher than the number based model. The rationale of the superiority is similar to the discussion above. When facing new agents and new transactional data, there is no way for the agents in the number based model to know which agent is malevolent and which one is benevolent. They have to interact with the new agents first even these agents are all malevolent agents. They need to use failures to build the knowledge about the agents. The agents of the proposed framework are much better for adapting the new environment. The estimation algorithm provides a protection in advance. However, the number based model does have its advantage. Especially in a familiar environment, the performance of the number based model is better than the proposed framework. This is due to the complexity of the STL algorithm and the trust estimation algorithm. Both algorithms take much time to learn and to process the transactional data. The number based model does not have such complexity; all the operations are simply adding 1 or subtracting 1 from the numeric trust value. The maintenance of the trust value is also simple and direct. Thus, in a simple and restricted virtual society, the population of the agents is limited. The number based model or its improved variations are good choice. In an internet based virtual society, the heterogeneity of agents is high or the requirements of adaptation for agents are high. The framework in this thesis is better choice.

From the results in the previous section, the reputation is proving to be an important complementary and improvement for the computational trust. In the result of Figure 6.7, the reputation enhanced framework greatly increases the rate of successful transaction. The biggest difference appears at the time of 9000 seconds, the difference reaches 40. The rationale behind is simple. When agents depend only on themselves to judge the malevolence and benevolence, they might be stuck into two dilemmas: either they lack the interaction experiences due to their short life cycle, or the environment changes are so significant that agents cannot adapt in a short time. In these situations, reputation become critical factor that help make up for the deficiencies.

There is another advantage of the reputation mechanism in the proposed framework. The process of receiving reputation is unique and creative. The reputation is transmitted in the form of messages which follows the format of the Agent communication language standards. The core part of the message is the reputation value. This number is not directly used by the receiver agent. The agent digests the reputations by accepting the reputation value as one of the input dimension of the input vector.

The reputations then become one part of the STL algorithm. Even some agent sends incorrect reputation to the host agent. If such reputation does not lead to positive transaction results, the patterns against this agent is influenced and tend to generate some features that denies the service level of the agent.

The way of propagating reputation can also influence the performances of the agents. In this scenario, the point-to-point based inquiry is used as the default propagating method. The agent sends reputation only when it is inquired by some other agents. Such methods are appropriate for communities with middle size of agents. For bigger virtual societies, the observer propagation is the better choice because this method can greatly decrease the communications among agents. It is activated only when the registered topic or registered agent have some state changes. The message is only transmitted between the subscribers and the subject owners. For extremely small society, the broadcasting propagation is the best choice. In the small society, there are only a few agents. The host agent knows all the other agents. When the host agent has any interactions, it can report the reputation generated to all the agents it knows. This is good for small society with high real-time or high responsiveness requirement.

6.5.2 A comparison between the neural trust model and the traditional trust models

The previous section analyzed the rationale behind the testing scenarios. In this section, the proposed neural trust model is compared with the existing traditional trust models. The result of the comparison is listed in table 6.7. There are 7 existing models which are listed to be compared with the neural trust model. They are TRAVOS (Teacy et al., 2006), FIRE (Huynh et al., 2004), ReGreT (Sabater and Sierra, 2002), Referal (Yu and Singh, 2000), Sporas (Zacharia et al., 1999), Marsh (Marsh, 1994*a*), and eBay (Ebay, 2012). The comparison is carried out from 7 selected aspects: Architecture, Trust domain semantics, Belief representation, Trust learning, Decision support, Reputation generation, and Propagation mechanism. Each aspect reflects one facet of strength or weakness of the model. So the synthesis from different aspects can show obvious advantages of the proposed model.

Model Name	Architecture	Trust Domain and Semantics	Belief Representation	Trust Learning	Decision Support	Reputation Generation	Propagation Mechanism
Neural Trust Model	Distributed	Ontological dimension	Patterns	Improved SOM based Neural Networks	Patterns differences reflect confidence	Pattern differences merged	Point-to-point, Broadcast, Observer
TRAVOS (Teacy et al., 2006)	Distributed	Applied to single domain	Probability (Bayesian framework)	N/A	Measure confidence in its value of trust	Weighted average of individual opinion	N/A
FIRE (Huynh et al., 2004)	Distributed	Ontology dimension	Number	N/A	reliability rating that reflects the confidence	Weighted accumulation of acquaintances	N/A
ReGreT (Sabater and Sierra, 2002)	Distributed	Applied to single domain	Number	N/A	Set threshold for the reliability	A weighted mean of the impressions' rating factors	N/A
Referal (Yu and Singh, 2000)	Distributed	Applied to single domain	Number	N/A	an upper and a lower threshold for trust	a total belief obtained from the original witnesses	Testimony propagation algorithm
Sporas (Zacharia et al., 1999)	Centralized	Applied to single domain	Number	N/A	A threshold is set for the reputation value	the weighted average of non-negative values	Directed graph representing the rating path
Marsh (Marsh, 1994*a*)	Distributed	Applied to single domain	Number	N/A	A threshold is set for the trust value	A weighted accumulation from agents	N/A
eBay (Ebay, 2012)	Centralized	Applied to single domain	Number	N/A	No threshold, judged by human users	Rated by human user	N/A

Table 6.7: Neural model vs. Existing models - Comparison between the neural trust model and the traditional trust models.

- Architecture: 6 of 8 models are distributed models. Only eBay (Ebay, 2012) and Sporas (Zacharia et al., 1999) are centralized models. Distribution is a natural feature that a multi-agent system should possess. Agents distributed over different nodes of network across various geographical zones. Trust models should also follow the distributed deployment of agents. The neural trust model is purely distributed in that every agent has its own STL and STE algorithm embedded so that they can do trust related activities independently. There is no central repository for trusts or reputations. The trust is generated individually and the reputation is aggregated from acquaintances.

- Trust domain and semantics: majority of the models are designed for single domain (6 of 8). Only the neural trust model and FIRE (Huynh et al., 2004) model considers the domain difference toward the involving objects. The neural trust model divides transaction data into dimensions according to their belonging domain. The dimensional data then is feed to the STL algorithm as multi-dimensional input. The precondition of two agents starting trust learning is that they both have consensus on ontology about their transactions.

- Belief representation: 6 models use number to represent the belief of trust, the TRAVOS (Teacy et al., 2006) model use probability to represent trust and its related confidence value, the neural trust model use pattern to represent trust. The number based models are inflexible and unable to reflect sophisticated changes on belief. It is also unable to differentiate trust from different domains. The probability based is better to reflect the fuzzy characteristic of belief. However, it is still difficult to show the domain difference using probabilities. The neural trust model uses patterns to represent trust. A pattern has abundant detail to map the sophistication of the belief. Furthermore, patterns are a simulation that mimics the mechanism of belief in human brain. It can successfully deal with the problem of different domains.

- Trust learning: All of the existing models do not support learning trust from previous interactions or transactions except eBay model (Ebay, 2012) which is human user based. They only evaluate and record the outcome of the interaction in a numbered or probabilistic way. They are poor to deal with the uncertainty

in interactions or transactions; they cannot quickly response to a fast changing environment. The neural trust model takes the trust recognition problem as a learning problem and learns trust using improved SOM based neural networks. This makes agents flexible enough to different domains and robust enough to sudden outside changes.

- Decision support: Threshold of trustworthiness decision making is common way of helping agents to decide actions. All models use threshold to control the decision making. The existing models set number as threshold, if the trust or reputation value supersedes the threshold. A trustworthy decision will be made. The neural trust model also uses threshold. However, the threshold is generated from the difference of estimated pattern and macro pattern. The value is between -1 to 1. The value is guidance to trustworthy decision or untrustworthy decision.

- Reputation generation: majority of the models (7 of 8) use weighted average or weighted accumulation to calculate the reputation value. The neural trust model takes a way to compare the target agent pattern with the global macro pattern. Then the result is accumulated to form the reputation. The advantage of adding the additional step is that the added pattern considers the difference between macro issues and a specific issue; it reflects difference between the general belief and the occasional idea. The weighted average or accumulation can not reveal such delicate details.

- Propagation mechanism: 3 0f 8 models never consider the problem of reputation propagation. The Referal model (Yu and Singh, 2000) designs a testimony propagation algorithm and the Sporas model (Zacharia et al., 1999) designs a directed graph representing the rating path. The proposed neural trust model designs three forms of propagation: point-to-point, broadcast, and observer. The idea of the design is coming from network protocols and design patterns. The advantage of three forms' propagation is that they can fit for different situations and is easy to implement. In addition, the decaying rate along the chain of reputation is also contribution of the new model.

6.5.3 Limitations

The model proposes a new way to learn and estimate trust from the agent transactions. The intelligent agents can adapt to the constantly changing environments. However, the model does have some limitations due to its way of designing and implementation. The first limitation is its relatively inferior performance comparing to some statistical or numeric models. The limitation is due to the drawback of neural network based algorithms. Much time is cost to exciting neighbouring neurons and update synaptic weight repeatedly. A good thing is that the usage of the model now is not real-time so that the agent has plenty of resources and time to deal with transactional data at each time. The other limitation is the way of input data. The data needs to be transformed into normalized dimensional data. The dimensions are so diversified in complex e-commerce environments. Even with different user, he or she may have different dimensions to evaluate the outcome of the transaction. There still future research to be done on the areas of semantic trust learning and e-commerce evaluation ontologies.

One of the major drawbacks of neural network computation, including the SOM algorithm, has been its computational complexity. It is computationally expensive. Typically, the computational complexity grows as $O(N^2)$ with number of terms N. Training instances are often presented multiple times and network performance is achieved only after gradual modification of network connection/link weights. Our experience in adopting SOM in several mid-size information visualization and categorization projects (10-100 MBs, several to hundreds of thousands of abstract-size documents) confirmed this general observation. The computational complexity of the SOM algorithm has rendered it infeasible for large-scale applications (1-10 GBs, millions of documents, e.g., the entire searchable Internet WWW homepages). In order to improve the scalability of the SOM approach to textual classification, a more efficient algorithm is needed.

The computational complexity of neural networks including the SOM based algorithms is their major drawbacks because the networks need to process thousands of training instances by multiple times. The network performance is delayed by gradual modification of the synaptic weights. The above experiments also show that the STL and STE (SOM based algorithm) obey this general observation. In large scale real-time application, the STL and STE are infeasible under the current effectiveness. However,

the usage of the STL and STE now is not used continuously or real-time. Basically, they are applied once after each round of agent transactions. The agent has sufficient resource and time to learn trust and estimate trustworthiness of the target agent in such circumstance.

Generally speaking, the neural trust model proposes a new systematical model to represent, generate, and exploit computational trust. The most significant advantage of the model is the idea of trust learning and the algorithm of trust learning. Looking trust as a machine learning problem also decides that the belief should be represented as patterns. Such representation endows more ability to record delicate details of the interactions and simulates the trust in human brain. The model also proposes a new pattern based trustworthiness decision making algorithm (trust estimation algorithm). The model designs a reputation propagation mechanism to support the generation and propagation of reputations. The mechanism designs three types of reputation propagation paths: Point-to-point based inquiry, broadcasting based propagation and observer. The reputation is produced as the difference between macro pattern and agent pattern.

6.6 Summary

This chapter mainly focuses on describing the experimental environment, test bed design and implementation, evaluation data preparation, evaluation configuration, and evaluation result. The chapter first describes the test bed design and implementation, shows the requirements, architecture and components. The rationale behind the design is also discussed in detail. Then the chapter introduces the experimental preparations including environment configuration and data preparation. The chapter also presents the results through two types of comparisons. One is to compare the proposed neural trust model with a simple number based trust model; the other one is compare the reputation based neural trust model with a non-reputation based neural trust model. The experimental result can support the argument that the neural trust model can have better quality and performance than the simple number based trust model. The results also states the importance of the reputation propagation in a multi-agent system. The chapter also discusses the results from the comparison and analyze some core design considerations and limitations.

7

Conclusion and Further Work

7.1 General summary

The thesis carries out research on the computational trust and reputation for the multi-agent systems. It reviews the background theories for intelligent agents and multi-agent systems. It investigates the state of art on the mechanisms, algorithms and models of trust and reputation in computer science. The features of these models are compared from several typical aspects and the advantages and disadvantages of these models are also analyzed and discussed in detail. Then the thesis analyzes the definition of the computational trust, defines the concept from a unique viewpoint, compares the computational trusts with human trusts, and discovers the key factors hidden behind the definition. Through the insight to the computational trust gained in the profound analysis, the thesis proposes a complete model named "Neural Trust Learning and Estimation Model" that systematically supports all trust and reputation related activities include learning trust, estimating trust, making trustworthiness decision, generating reputation, and propagating reputation. In this model, the thesis innovates the traditional Self Organizing Map (SOM) and creates a SOM based Trust Learning (STL) algorithm and SOM based Trust Estimation (STE) algorithm. The STL algorithm solve the problem of learning trust from agents' past interactions and the STE solve the problem of estimating the trustworthiness with the help of the previous patterns. For transforming the computational trust into a social influential mechanism, the thesis also proposes a multi-agent reputation mechanism for generating and propagating the reputations. The mechanism exploits the patterns learned from STL algorithm and

generates the reputation of the specific agent. Three propagation methods are also designed as part of the mechanism to guide path selection of the reputation. In order to evaluate the viability and effectiveness of the model and the algorithms/mechanism in the model, the thesis designs and implements a test bed that is based upon the Java Agent Development Environment platform. A simulated electronic commerce scenario is used with lots of transactional data to test whether the proposed framework works well. The traditional arithmetic based trust model is compared with the proposed Neural Trust Learning and Estimation Model. The experimental results prove the superiority of the proposed model. The quality and performance of the transactions have been improved in experiments implemented with the proposed model. The experimental results also state the important influence of applying reputation mechanism. Some design considerations and rationale behind the algorithms are also discussed based on the experimental results.

7.2 Contributions

The thesis analyzes the idea of learning trust, proposes a systematical model for trust and reputation, proposes the STL algorithm and STE algorithm, designs the reputation generation and propagation mechanism, and implement a test bed. These significant contributions are listed and explained in brief as follow:

- New ideas proposed and analyzed: The thesis believes that the computational trust is a combinatory problem of machine learning and decision making. The form of representation is patterns that are emerged from information collected during agent interactions. The thesis also points out that the trustworthiness and untrustworthiness are actually decision making problems. Most of the traditional computational trust models are arithmetical or statistical models which deem the trust and reputation as a numeric value generated from delicate equations.

- A Systematic model: the thesis proposes a trust and reputation model named "Neural trust learning and estimation model". This model establishes a complete framework to accomplish trust related activities. It depicts the relationship among algorithms and mechanisms. It defines the processes of trust learning, estimating and reputation generating, propagation.

- Algorithms for learning and estimating trust: two algorithms are proposed. One is SOM based Trust Learning algorithm and the other is SOM based Trust Estimation algorithm. Both algorithms modify the original Self Organized Map to produce patterns suitable for representing trust. The estimation algorithm helps agents make trustworthiness decision according to its memorized patterns.

- Reputation propagation mechanism: the thesis proposed a reputation propagation mechanism to support the generation and propagation of reputations. The mechanism designs three types of reputation propagation paths: Point-to-point based inquiry, broadcasting based propagation and observer. The reputation is produced as the difference between macro pattern and agent pattern. An equation for calculating the reputations is proposed and issues such as decaying along the propagation are also modeled in it.

- Application: In order to evaluate the model, a computational trust test bed is designed and implemented. The test bed is built upon Java Agent Development Environment platform and it constructs a simulated electronic market which contains many autonomous agents divided into different roles. The test bed not only can be used in this thesis related experiment, but it also can be used to test the other types of agent trust models.

7.3 Future work

The thesis involves the research on computational intelligence and machine recognition, the development in these areas is only a start comparing to other disciplines with hundreds years of history. There are still more truths to be discovered and more problems to be solved. Although a lot of efforts have been put into the research for this thesis, there are still plenty of areas that needs to be explored in future studies.

Due to the scarcity of commercial transaction data from large scale e-commerce web sites (Electronic commerce sites like eBay already has its reputation systems supporting their transactions. To test the proposed model in such a site, it is necessary to substitute their original reputation system and training their millions of user to use it. This is difficult and could lead to considerable cost. A better way to make use of the open API of the site (presume they have), take the proposed model as a site plugin and let part

of the users uses and evaluates the model based trust system. Both ways needs the authorization and strong support from the site owner), the experiment in the thesis is only based on simulated scenario of automated electronic commerce. The experimental data is also generated from the rule based computer programs. In future developments, if it is possible, it is better to find commercial web sites as the a research partner and test the model in a real environment. The autonomous agents can represent the human user to do transactions, and the human user can set their evaluations as the input to the computational trust models.

In Chapter 4, the domains and dimension of the computational trust has been analyzed and discussed. The model also takes domain based dimensional data as input to the learning algorithm. However, even the agent knows the domain and dimension; the semantics behind the name is not unified. This could lead to problem of duplicating or misunderstanding the name of domain or dimension. Fox example, the product name or domain name in different languages may refer to the same object. Additionally, the scarce of a unified domain hierarchy is an obstacle of building large scale multi-agent societies because there might be domain name conflicts and domain structure conflicts that block extended interactions cross systems. Thus, a unified ontology structure defining domains and dimensions, and a semantic web based implementation of the structure are the prerequisites for constructing large multi-agent societies.

Multi-agent organization is a research topic in multi-agent system. It refers to the organization methods of the roles, relationships and authority structures in a multi-agent system. The thesis discusses the reputation generation and propagation mechanisms. Trust and reputation can actually be the adhesion of the organization or society. In future research, it is possible to implant the computational trust and related models into the organization to see the how trust influence the formation and structure of the organization. Meanwhile, the organization theory, the organization structure can change the way that reputation propagates or create new forms of propagation mechanisms.

References

Adolphs, R. (2002), 'Trust in the brain', *NATURE NEUROSCIENCE* **5**(3), 192–194. 43

Alonso, E, D. M. K.-D. L. M. and Noble, J. (2001), 'Learning in multi-agent systems', *The Knowledge Engineering Review* **16**, 277–284. 20

Ana Lilia Laureano Cruces, F. D. A. (2000), 'Reactive agent design for intelligent tutoring systems', *Cybernetics and Systems: An International Journal* **31**(1), 1–47. 13

Barber, K. S. and Kim, J. (2003), 'Soft security: Isolating unreliable agents from society', *LECTURE NOTES IN COMPUTER SCIENCE* **2631**, 224–234. 6

Bates, J. (1994), 'The role of emotion in believable agents', *Communications of the ACM* **37**(7). 11

Bates, J., Loyall, A. B. and Reilly, W. S. (1992), 'An architecture for action, emotion, and social behavior', *Technical Report CMU-CS-92-144, School of Computer Science, CMU* . 11

Bellifemine, F., Caire, G. and Greenwood, D. (2007), *Developming multi-agent systems with JADE*, John Wiley and Sons. 30, 103, 106, 107, 108

Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F. and Mylopoulos, J. (2004), 'Tropos an agent-oriented software development methodology', *AUTONOMOUS AGENTS AND MULTI AGENT SYSTEMS* **8**(3), 203–236. 22

Briot, J. P. and Gasser, L. (1998), 'Agents and concurrent objects', *IEEE CONCURRENCY* **6**(4), 74–77. 22

Castelfranchi, C. Dignum, E. J. C. M. T. J. (1999), 'Deliberate normative agents: Principles and architecture', *LECTURE NOTES IN COMPUTER SCIENCE, Springer-Verlag* **1757**, 364–378. 13

Castelfranchi, C. and Falcone, R. (2000), 'Trust is much more than subjective probability: Mental components and sources of trust.', *Proc. of the 33rd Hawaii International Conference on System Sciences* **6**. 52

Cofta, P. (2007), *Trust, Complexity and Control: Confidence in a Convergent World*, John Wiley and Sons. 37

Cohen, P. R. and Levesque, H. J. (1990), 'Intention is choice with commintment', *Artificial Intelligence* **42**(3). 12

coordinating team, F.-O. (2012), 'Fipa-os developers guide'. http://citeseer.ist.psu.edu/477218.html. 30

Coulouris, G., Dollimore, J. and Kindberg, T. (2000*a*), *Distributed systems: concepts and design*, Addison-Wesley Longman Publishing Co., Inc. 7

Coulouris, G., Dollimore, J. and Kindberg, T. (2000*b*), *Distributed systems: concepts and design*, Addison-Wesley Longman Publishing Co., Inc. 7

de Mantaras, R. L. (2001), 'Case-based reasoning', *Machine Learning and Its Applications Lecture Notes in Computer Science* **2049**, 127–145. 21

DeGroot, M. and Schervish, M. (2002), *Probability and statistics (3rd edn.)*, Addison-Wesley. 25

Dempster and Arthur, P. (1968), 'A generalization of bayesian inference', *Journal of the Royal Statistical Society* **30**, 205–247. 25

E., R. C. and Yuhui, S. (2007), *Computational intelligence: concepts to implementations*, Morgan Kaufmann. 7, 66, 73

Ebay (2012), 'Ebay web site'. http://www.ebay.com. 23, 28, 121, 130, 131, 132

FIPA (n.d.), 'Fipa acl message structure specification'. 105, 111

Fisher, M. and Wooldride, M. (1997), 'On the formal specification and verification of multi-agent systems', *International Journal of Cooperative Information System* **6**(1). 18

Fukuyama, F. (1996), *Trust: The Social Virtues and the Creation of Prosperity*, Touchstone Books. 37

Gehao, L., Joan, L., Shaowen, Y. and Jim, Y. (2009), 'A comparison of java-based multi-agent system development platforms', *The Open Information Science Journal* . 106, 107, 108

Grosz, B. and Kraus, S. (1996), 'Collaborative plans for complex group actions', *Artificial Intelligence* **86**(2), 269–357. 15

Hannemann, J. Kiczales, G. (2002), 'Design pattern implementation in java and aspectj', *ACM SIGPLAN NOTICES* **37**(11), 161–173. 91

Haykin, S. (1999), *Neural Networks: A Comprehensive Foundation*, Prentice-Hall. 64

Hewitt, C. (1985), 'The chanllenge of open systems', *Byte* **10**(4), 223–242. 7

Huberman, B. A. and Hogg, T. (1995), 'Communities of practice: Performance and evolution', *Computational & Mathematical Organization Theory* **1**(1), 73–92. 19

Huynh, T. D. (2006), Trust and Reputation in Open Multi-agent Systems., PhD thesis, Schools of Electronics and Computer Science, University of Southampton. 8, 9, 24, 25, 28

Huynh, T. D., Jennings, N. R. and Shadbolt, N. R. (2004), 'Fire: An integrated trust and reputation model for open multi-agent systems', *ECAI* **16**, 18–22. 2, 3, 130, 131, 132

Huynh, T. D., Jennings, N. R. and Shadbolt, N. R. (2006), 'An integrated trust and reputation model for open multi-agent systems', *Autonomous Agent and Multi-Agent Systems* **13**, 119–154. 9, 24, 25, 28

Jeffrey S. Rosenschein, G. Z. (1994), *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*, The MIT Press. 19

J.Ferber (1999), *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*, Addison-Wesley. 8

Josang, A., Ismail, R. and Boyd, C. (2007), 'A survey of trust and reputation systems

for online service provision', *Decison Support System* **43**, 618–644. 26

Josang, A. and Pope, R. S. (2005), 'Semantic constraints for trust transitivity', *APCCM 2005, University of Newcastle* . 29

K. Eric Drexler, S. M. M. (1988), 'Incentive engineering for computational resource management.', *The Ecology of Computation* **In B. A. Humberman**, 231–266. 19

Kinny, D., Georgeff, M. and Rao, A. (1996), 'A methodology and modelling technique for systems of bdi agents', *LECTURE NOTES IN COMPUTER SCIENCE* **1038**, 56–71. 13, 22, 38

Koehn, D. (2003), 'The nature of and conditions for online trust', *JOURNAL OF BUSINESS ETHICS* **43**, 3–19. 79

Kohonen, T. (1998), 'The self-organizing map', *Neurocomputing* **21**, 1–6. 64

Kraus, S. (1997), 'Negotiation and cooperation in multi-agent environments', *Artificial Intelligence Journal, Special issue on Economic Principles of Multi-Agent System* . 15

Laboratory, G. (2005), Agent system development hands-on exercise, easss-05 (7th european agent systems summer school), Technical report, Utrecht. 30

Labrou, Y. and Finin, T. (1997), 'A proposal for a new kqml specification', *TR-CS-9703, UMBC* . 14

Labrou, Y., Finin, T. and Peng, Y. (1999), 'Ieee intelligent systems and their applications', *IEEE INTELLIGENT SYSTEMS AND THEIR APPLICATIONS* **14**(2), 45–52. 14

Lenat, D. B. and Brown, J. S. (1984), 'Why am and eurisko appear to work', *Artificial Intelligence* **23**, 269–294. 19

Levesque, H. J. (1990), 'All i know: A study in autoepistemic logic', *Artificial Intelligence* **42**, 263–309. 15

Liviu Panait, S. L. (2005), 'Cooperative multi-agent learning: The state of the art', *Autonomous Agents and Multi-Agent Systems* **11**(3), 387–434. 19

Longman (2011), 'Longman online dictionary'. http://www.ldoceonline.com/dictionary/trust$_2$.36

Luck, M. and d'Inverno, M. (1995), 'A formal framework for agency and autonomy', *In Proc. of the First International Conference on Multi-Agent Systems, AAAI Press* . 18

Luke, T. W. (2006), Agent-based trust and reputation in the context of inaccurate information sources, PhD thesis, Schools of Electronics and Computer Science, University of Southampton. 9, 24, 25, 30

Luke Teacy, W. T., Patel, J., Jennings, N. R. and Luck, M. (2005), 'Coping with inaccurate reputation sources: Experimental analysis of a probabilistic trust model', *AAMAS?05* . 9, 24, 25

Marsh, S. (1994*a*), Formalising Trust as a Computational Concept, PhD thesis, Department of Mathematics and Computer Science, University of Stirling. 9, 24, 130, 131

Marsh, S. (1994*b*), 'Trust in distributed artificial intelligence', *LECTURE NOTES IN COMPUTER SCIENCE* **830**, 94. 2, 3, 48

Mayer, R., Davis, J. H. and Schoorman, F. (1995), 'An integrative model of organizational trust', *Academy of Management Review* **20**(3), 709–734. 37

McKnight, D. H. and Chervany, N. L. (1996), The meanings of trust, Technical report, University of Minnesota? 52

Merriam-Webster (2011), 'Merriam-webster online dictionary'. http://www.merriam-webster.com/dictionary/trust. 36

Minsky, N. H. and Murata, T. (2004), 'On manageability and robustness of open multi-agent systems', *LECTURE NOTES IN COMPUTER SCIENCE* **2940**, 189–206. 7

Mollering, G. (2005), 'The trust/control duality: An integrative perspective on positive expectations of others', *Int. Sociology* **20**(3), 283?305. 52

Moore, R. (1990), 'A formal theory of knowledge and action', *In formalizing common sense, Ablex publishing corperation* pp. 319–358. 12

Moreno, Y., Nekovee, M. and Pacheco, A. F. (2004), 'Dynamics of rumor spreading in complex networks', *PHYSICAL REVIEW - SERIES E-* **69**, 066130. 55

Nwana, H., Ndumu, D., Lee, L. and Collis, J. (1999), 'Zeus: a toolkit and approach for building distributed multi-agent systems', *Proceedings of the Third International Conference on Autonomous Agents (Agents?99)* pp. 360–361. 30, 106, 107, 108

Ourston, D. and Mooney, R. J. (1994), 'Theory refinement combining analytical and empirical methods', *Artificial Intelligence* **66**(2), 273–309. 20

Oxford (2011), 'Oxford online dictionary'. http://oxforddictionaries.com/definition/trust. **URL:** *http://oxforddictionaries.com/definition/trust* 36, 45, 46

Pazzani, M. and Kibler, D. (1991), 'The utility of knowledge in inductive learning', *Machine Learning* **9**(1), 57–94. 20

Pazzani, M., Merz, C., Murphy, P., Ali, K., Hume, T. and Brunk, C. (1994), 'Reducing misclassification costs', *Proceedings of the Eleventh International Conference on Machine Learning* pp. 217–225. 20

Ralph, A. (1999), 'Social cognition and the human brain', *Trends in Cognitive Sciences* **3**(12), 469–479. 43

Resnick, P., Zeckhauser, R., Swanson, J. and Lockwood, K. (2006), 'The value of reputation on ebay: A controlled experiment', *EXPERIMENTAL ECONOMICS* **9**(2), 79–101. 2, 3

Ritter, H. (1999), 'Self-organizing maps on non-euclidean spaces', *Kohonen Maps* . 73

Rothkopf, M. H. (2003), 'The future of e-markets', *J Econ Lit 2003* **41**(1), 214. 7

S., R. A. and P., G. M. (1991*a*), 'Deliberation and intentions', *Technical Notes 10, Australian Artificial Intelligence Instistue* . 12

S., R. A. and P., G. M. (1991*b*), 'Modeling rational agents within a bdi-architecture.', *Proceedings of the Second International Conference on Principles of KRR, Morgan Kaufmann* . 18

Sabater, J. (2003), Trust and Reputation for Agent Societies, PhD thesis, Universitat Autnoma de Barcelona. 9

Sabater, J. and Sierra, C. (2001), 'Regret:areputationmodel for gregarious societies', *In Fourth workshop on deception fraud and trust in agent societies* pp. 61–70. 9, 24

Sabater, J. and Sierra, C. (2002), 'Social regret, a reputation model based on social relations', *ACM SIGecom Exchanges - Chains of commitment* **3**(1). 2, 130, 131

Sandra M. Hedetniemi, Stephen T. Hedetniemi, A. L. L. (2006), 'A survey of gossiping and broadcasting in communication networks', *Networks* **1988**(4), 319349. 89

Shoham, Y. (1992), 'Agent-oriented programming: An overview and summary of recent research', *In Proc. of Artificial Intelligence* . 11

Shoham, Y. (1993), 'Agent-oriented programming', *Artificial Intelligence* **60**, 51–92. 11

Simon, H. A. (1996), *The Science of Artificial (3rd edn)*, The MIT Press. 7

Simon Parsons, M. W. (2002), 'Game theory and decision theory in multi-agent systems', *Autonomous Agents and Multi-Agent Systems* **5**(3), 243–254. 19

Singh, M. P. (1994), 'Multiagent systems: A theoretical framework for intention as, know-how, and communications', *Lecture Notes in Artificial Intelligence* **799**. 12

Smith, R. G. (1980), 'The contract net protocol: High-level communication and control in a distributed problem solver.', *IEEE Transaction on Computrers* **12**. 16

Subrahmanian, V. S., Bonatti, P., Dix, J., Eiter, T., Kraus, S., Ozcan, F. and Ross, R. (2000), *Heterogeneous Agent Systems*, The MIT Press. 8, 9

Sztompka, P. (2000), *Trust: A Sociological Theory (Cambridge Cultural Social Studies)*, Cambridge University Press. 28

Taobao (2011), 'Chinese taobao web site'. http://www.taobao.com.
URL: *http://www.taobao.com* 121

Teacy, W. T., Patel, J., Jennings, N. R. and Luck, M. (2006), 'Travos: Trust and reputation in the context of inaccurate information sources', *AUTONOMOUS AGENTS AND MULTI AGENT SYSTEMS* **12**(2), 183–198. 2, 3, 130, 131, 132

Team, F.-O. D. (2011), 'Fipa-os developers guide'. http://citeseer.ist.psu.edu/477218.html. 106, 107, 108

Werger, B. B. (1999), 'Cooperation without deliberation: A minimal behavior-based approach to multi-robot teams', *Artificial Intelligence* **110**(2), 293–320. 18

Wikipedia (2012), 'Belief in wikipedia'. http://en.wikipedia.org/wiki/Belief. 48

Wooldridge, M. J. (2000), *Reasoning about rational agents*, The MIT Press. 22

Wooldridge, M. J. (2002), *An introduction to multiagent systems*, John Wiley and Sons. 7, 11, 13

Wooldridge, M. and Jennings., N. R. (1994), 'Towards a theory of cooperative probelm solving.', *In Proc. of Modelling Autonomous Agent in a Multi-Agent World (MAAMAW-94)* . 15

Yu, B. and Singh, M. P. (2000), 'A social mechanism of reputation management in electronic communities', *Cooperative Information Agents IV - The Future of Information*

Agents in Cyberspace, Lecture Notes in Computer Science **1860/2000**, 355–393. 2, 3, 130, 131, 133

Yu, B. and Singh, M. P. (2002), 'An evidential model of distributed reputationmanagement', *In Proceedings of first international joint conference on autonomous agents and multi-agent systems, ACM Press* **1**, 294–301. 24, 25

Yu, B. and Singh, M. P. (2003), 'Searching social networks', *In Proceedings of the second international joint conference on autonomous agents and multiAgent systems (AAMAS), The ACM Press* pp. 65–72. 24, 25

Zacharia, G. (1999), Collaborative reputation mechanisms for online communities, Master's thesis, Massachusetts Institute of Technology. 9, 23

Zacharia, O., Moukas, A. and Maes, P. (1999), 'Collaborative reputation mechanisms in electronic marketplaces', *PROCEEDINGS OF THE HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES* **32**, 300. 2, 3, 130, 131, 132, 133

Zhongzhi, S., Tao, W., Qijia, T. and Hui, T. (1994), 'Mape: Multi-agent processing environment.', *Proceedings of PRICAI'94, International Academic Publisher* . 13

Zhou Wen-li, W. X.-f. (2006), 'Survey of p2p technologies', *Computer Engineering and Design* **01**(22). 87

Declaration

I herewith declare that I have produced this paper without the prohibited assistance of third parties and without making use of aids other than those specified; notions taken over directly or indirectly from other sources have been identified as such. This paper has not previously been presented in identical or similar form to any other British or foreign examination board.

The thesis work was conducted from June 2006 to Aug 2011 under the supervision of Prof. Joan Lu at University of Huddersfield.

Huddersfield,UK