The Department of Computer Science
The University of Auckland
New Zealand

A Formal Security Modeling and Analysis in B2B e-commerce

Han Zhang

July 2006

Supervisors:

Clark Thomborson and Gerald Weber

A thesis submitted in partial fulfillment of the requirements of Doctor of Philosophy in Computer Science

The University of Auckland

Thesis Consent Form

This thesis may be consulted for the purpose of research or private study provided that due acknowledgement is made where appropriate and that the author's permission is obtained before any material from the thesis is published.

I agree that the University of Auckland Library may make a copy of this thesis for supply to the collection of another prescribed library on request from that Library; and

1. I agree that this thesis may be photocopied for supply to any person in accordance with the provisions of Section 56 of the Copyright Act 1994.

   Or

2. ~~This thesis may not be photocopied other than to supply a copy for the collection of another prescribed library.~~

   (*Strike out 1 or 2*)

Signed: .....

Date: .....

Created: 5 July 2001

Last updated: 9 August 2001

Abstract

Despite the flourishing research on formal modeling and analysis of privacy and authentication issues in E-commerce, little research concentrates on the possible security risk due to business logic specification. In E-commerce systems, an aspect of this logic is to promise fairness. As the feature ensuring parties conduct their business to their mutual moral standards, fairness is one of the paramount features for E-commerce payment systems. In this case study of the AARN payment system, we apply form-oriented analysis to formally model the simple business logic behind this way of arranging payment. The model solves a fair exchange issue for security purposes at the business logic level, which changes further design and implementation for the payment system. It is the first time that Data Type Interchange Model diagram and other models in form-oriented analysis method have been applied in security analysis. This form-oriented analysis method helps designers not only on security analysis, but also on understanding and communication between business experts and software designers.

Acknowledgement

First of all I would like to thank Barry Dowdeswell for without him I would not have had the opportunity to work with AARN on this research topic. I have known Barry since the summer school project in my postgraduate diploma study. He is a great tech director and also a perfect team manager; as far as I know he was organizing serval projects at the same time, meeting clients to expand the market, contacting the university for co-operating the research projects; meanwhile he still had time to help me on my writing. He works like a superman and I know he spent lots of his spare time for me and many other students, but I do not know how to express my thanks for him – never more than enough.

The greatest thanks for Professor Clark Thomborson and Dr Gerald Weber, my thesis supervisors. They gave me thousands of ideas, suggestions and the most important, encouragement. I learned not only academic research skill, but also attitudes for science and technology from them. I credit them for teaching me a critical and logical thought, a clear and strict description, and a modest as well as challenging attitude.

Also, I would like to thank William Zhu, Jasvir Nagra, Stephen Drape, Anirban Majumdar, and Barbara Thomborson. I had fun working with William and am also impressed by his unbelievable productivity. William encouraged me on the submission of my first paper and worked with me to solve the mathematical expression problems in the paper. Jasvir helped me on the proof reading and also gave me useful comments. Furthermore, without the thesis pattern he designed, I can not imagine how messy a style my thesis will

be. Stephen helped me on constructing the mathematical equations and constraints, and proof reading. Anirban discussed with me on many security issues which broadened my insights. Barbara helped me modified a lot of grammar mistakes and wrong expressions. I would also like to thank other members in our security systems group for their useful hints and friendly discussions. I especially enjoyed the paper discussions in regular group meetings.

Contents

1 Introduction

Following the over 20% increase of e-commerce in North America, internet fraud has led to around US$2.8 billion revenue lost in 2005 [11]. Though e-merchants have expensed huge amounts on security, fraud is still a tough problem for e-commerce [25]. Difficulties in anticipating novel frauds have made research in security measures very important for e-commerce.

Security measures in e-commerce have various methods and strategies for different purposes. Different methods are suitable for specific situations, but there is no overall suitable method to foil all internet frauds for every situation. We define the possibility of a fraud as a **risk**. In certain e-commerce-related areas, such as networking, data transfer and data storage, researchers applied scanning and testing methods, modeling analysis to detect potential risks. For example, the model of discretionary access control model, Mandatory Access Control (MAC) model, Role-based Access Control (RBAC) model, and Access control Tasks/Workflow are used to analyze the access control functions [22]. By applying these models, designers can offer different features and describe the perspective from different points of view to understand, clarify and solve the security problems.

However, despite sufficiently developed modeling in specific areas, more work is still needed to improve the models for describing abstract business logic. Such models should

not focus on the detail of the implementation technologies, but rather on business logic and rules, or the relationship and interaction between the actors. As a new form of commerce, e-commerce business logic differs in some ways from traditional thinking. Twenty years ago, business people could not imagine building a business network without a lot of physical support. Neither could they understand building a "good" business only by registering to a powerful searching engine, such as Google. E-commerce has shown that as long as information providers can attract a sufficient population, they can be extremely successful. E-commerce can be lucrative in several areas, such as in advertising and sharing profits from other service providers, and more e-commerce models are emerging. As a result of this flourishing business, researchers of the security of e-commerce are looking to improve their secure model to be suitable for the latest technological or business development.

1.1 B2B E-commerce

E-commerce operates as either business to customer (B2C) or business to business (B2B). Within B2C there is a special subset of customer to customer (C2C) auction business, which we will not discuss in this thesis. Unlike the B2C e-commerce threats from huge credit risk issues, B2B e-commerce has made exciting progress [26, 24]. This is partially because B2B transactions occur between business people, who have chance to learn each other's trustworthiness – B2B makes customer relations even more important than before [1].

In e-commerce, organizations and business parties solve their security issues through various ways. Banks have built a credit-based risk assessment strategy and applied electronic-funds clearance systems to manage their financial risk from e-commerce. Large businesses employ specialists, such as IT auditors, to review their systems and advise on best practice [2]. Consumers are managing their risk by slowing down their participation in e-commerce, which makes previous expectations for the e-commerce look overly optimistic. A survey in 2002 illustrated that internet shopping is more riskier than offline

shopping [41]. The risk perceived by consumers negatively affected their online shopping, except on low-value items such as books [2]. Other surveys present similar conclusions that the low level of consumer trust is limiting the growth of e-commerce [36]. Online retail sales are still a small part of total retail sales volume [44].

Small businesses may be in the most complex position of all. On the one hand, they are not as trustworthy to customers as the big companies, especially in B2B e-commerce, which means their business growth is quite small. On the other hand, as long as small businesses can easily build up relations with their customers, they can combine their advances on flexibility in business with the convenient and speedy features of e-commerce. However, these business parties are the most likely to experience the negative impact of the slow trade payment process and to look for more efficient payment terms [7]. Unlike banks and larger business, small businesses cannot define proper e-payment systems by which they minimize their risks. Unlike consumers, they may be unable to afford to wait on the sidelines until mature e-commerce systems come out. They will have a hard time on finding a profitable niche market in such a well-established online marketplace [2]. Like large businesses, small businesses must maintain compatibility with all their business parties, so they probably cannot use out of the box software [2]. Unlike large businesses, however, small businesses have additional challenges in that they cannot afford to hire a specialist to audit their semi-customized e-commerce systems, nor can they afford the maintenance due to the changes in business requirements. For example, all suppliers of Wal-Mart are required to comply with the recently developed Applicability Statement 2 (AS2) for electronic data interchange (EDI) over the internet [6, 29]. A typical small business party, as a Wal-Mart supplier, cannot suddenly transfer all its EDI transactions of their e-business system to AS2. Such a switch would destroy their existing business-partner relationship with those companies that do not simultaneously adopt AS2. As a result, many of the smaller Wal-Mart suppliers may well continue to use a Value Added Network (VAN) or accept both VAN and AS2, as a cost-effective way to achieve compatibility with all EDI formats in use by their business parties [2].

Sufficient security analysis can help the business parties, especially the small busi-

nesses, to represent the potential risks before they actually implement the security measures. Furthermore, a clear and rigidly accurate analysis can improve the understanding of the business logic and rules.

1.2 Fair Exchange in E-commerce

As more business becomes involved in e-commerce, there are more concerns about the execution on exchanging process [4]. A very basic question is how to ensure the correct execution of the exchange. In a correct exchange, the execution must be ensured in the terminal situation, when either both the business parties successfully approach their goals, or both of the parties failed but both of them contain their items. For example in an e-commerce scenario, party A is going to exchange his *itema* to party B's *itemb*. In a correct exchange, the final result is party A has *itemb* and party B has *itema*, or party A still keeps *itema* and party B keeps *itemb*. In [3], Dr Asokan first defined fairness as the feature which keeps the e-commerce exchange as a correct one.

A fair exchange is a necessary promise to the business parties, as people still doubt the reliability of e-commerce. According to a survey by Cybersource, a majority of the customers suspect that the items which they are interested could finally purchase [11]. A fair exchange can resolve the problem and ensure the exchange, therefore is attractive to customers. B2C e-commerce has made progress on the fairness of the exchange. Online auction systems such as eBay and TradeMe have setup many policies to ensure the purchases on their system are reliable for customers. Moreover, in online digital media purchases, researchers apply the digital signatures method for providing fair exchange. In B2B e-commerce, however, the parties have not paid enough attention to fairness of the exchange. It might be because most of the business parties believe their contract can protect their benefits in a legal environment. Despite the protection of the legal environment, in a fair B2B exchange, the process itself is limited as a fair process. Therefore even if either of the parties is going to commit fraud in the process, the process will prevent them from doing so. The business can stop the fraud even before the legal process starts.

As a result, honest parties benefit from a fair exchange.

Another reason for the low concern of fair B2B exchange is that most of the B2B customers purchase large amounts of goods. This introduces in a delivery issue for the fair exchange. However, if we avoid discussing the delivery process of the purchase but focus on the payment process, we can apply fair exchange, particularly in the payment process. To analyze the fair exchange process, we utilize formal analysis in the modeling of the fair exchange.

1.3 Formal Analysis

Formal analysis uses mathematical representations of a concept to deduce and analyze the concept. It is the basis of rigorous proofs in scientific research. Formal analysis can be complex, as it must include adequate information to support the steps in the proof. In this paper we describe a novel formal analysis method of a security model which is simple and clear. We start by documenting the existing business logic of a set of business parties at the message-passing (EDI) level of detail. The end result of this documentation is called a Data Type Interchange Model, or DTIM, and a complementary User Message Model or UMM [15]. We then categorize the final possible states of the model as being either expected or unexpected, and either harmful or beneficial [12]. These determinations are made relative to the business we are modeling. For example, making a net profit on a business transaction is an expected-beneficial outcome in most situations. Not receiving payment within a billing cycle for goods that have been shipped would be considered an unexpected-harmful outcome [2].

After we have developed a DTIM/UMM model for all parties in a transaction, we analyze it for any potential unexpected-harmful states. If we find any unexpected-harmful states, in a final step of our method we diagnose the problem and suggest changes to the business logic of the party(s) to minimize chances of reaching such a harmful state [2].

Our method has a second use, of discovering security flaws before any business logic is actually implemented by a business party. The precondition for this use is that the

business logic of prospective business parties must be known to the DTIM level of detail or abstraction [2].

The novelty of our methodology stems from its use of DTIMs as modeling abstraction. Many other modeling abstractions are possible. A few have been explored extensively, and we will survey them later in this paper. For now we note that comparing modeling abstractions is like comparing apples and oranges in that the result of the comparison depends on what you want. It might be easier, or harder, to capture an existing business logic in another abstraction. Some abstractions will permit some forms of security analysis; and some abstractions will *not* support the type of state-analysis we have outlined above. Some abstractions are at a high level of detail, and thus will support a very precise analysis. It is generally more difficult to gain confidence in the accuracy of a high-detail model than in a simple model, since a small error in one part of a complex model can cause a very large change in its behavior [2].

1.4 Motivation

In this thesis, we use a case study to illustrate the function of our method, and to argue that it is an appropriate way to analyze the business logic of a small enterprise, the AARN Innovation Limited, on its application of B2B payment system. Business logic is the thinking and rationales for making business decisions, especially concerning operations. In our case study it concerns the essential steps of how the business is operated. Software engineers must understand the business logic very well before designing the system. When we discuss the business logic, the issues are in an early step of the requirement analysis and functional specification or prototyping [37]. The method should allow the software designers to communicate with the business experts and understand the functions of the business without any misunderstanding by using this method. The method must be very strict, formal and simple to understand for the business experts. The method also needs to be flexible and easy to extend so that it can fit the requirements of describing various business logic.

Our case study has a simple four-party e-payment scheme involving a vendor, a buyer, a trusted third-party intermediary, and a banking subsystem. Despite its apparent simplicity, this scheme contains a harmful-unexpected state. We believe, and we invite readers to test this for themselves when they learn the details of the scheme, that the security flaw is not discoverable except by formal analysis or intense thought. Careful thought is, of course, the hallmark of a competent security analyst or auditor. Most analysts will want to frame and explain their thoughts in a convenient formalism or abstraction, if that is possible. This is a third potential use for our method – as an expressive abstraction in which analysts can document a non-trivial security bug so that competent programmers can repair the bug [2].

In commerce systems, especially e-commerce applications, relatively little security analysis is done at the business logic level. Most analysis at this level is focused on detecting what we would call mistakes in the implementation of a set of business rules, rather than detecting mistakes in their design. Examples of implementation error are the improper configuration of a CGI server, or the choice of CGI as an implementation language in the first place [31]. Examples of the business logic error are frauds in the business process, such as unsuitable transactions and abuse for personal enrichment through deliberate misuse or misapplication [45].

Once appropriate business logic is made clear, we are reasonably assured that all is well – if the designer accurately understands the logic, and if the logic itself is not buggy. If the logic is not formally specified, as is often the case, then we must observe the behavior of the implemented system carefully, to determine whether or not the logic is doing what we had expected. It is scary enough to undertake such a debugging-by-observation only once, when a new business is launched. At that time we probably have many, and larger, risks of complete business failure than the operational discovery of a catastrophic bug in our business logic. However, even after our business is well established, some of our software systems are changing monthly, if not daily. It is always possible that a seemingly innocuous change in one part of a computing system will drive another part of the system into an unexpected and harmful state [2, 12]. This line of reasoning pushes us either

toward the use of a formal specification for business logic which can be analyzed and verified, or toward the use of a context-based "business rules" framework [2].

1.5 Related Works

Previous researchers of business logic modeling and analysis have approached several areas of the e-commerce business analysis issue. A role and task-based security model (R&T model) was applied to ensure a secure access to many different services through an application-based security framework. It is used and implemented in a multi-functional smartcard to ensure both the users' need for application-based security and their right to informed self-determination. The fundamental right of privacy is related to legal issues [38].

Another model based on phases in business processes and roles and interactions in each phase has a simple approach to understanding e-commerce business models. This model categorizes several typical business models and then analyzes the specific security requirements. It highlights potential threat scenarios and describes their solutions. This is a decomposition approach for e-commerce business models and its application to the systematic assessment of their security requirements [18].

Unified Modeling Language (UML) is also applied in security modeling processes. Based on role-based access control with additional support of constraints on authorization, UML provides a new direction for applications in specifying access control information and modeling this security issue in the design of an application [27]. This information can be used to automatically generate complete access control infrastructures. By applying UML security modeling, the researchers can improve productivity during the development of secure distributed systems and the quality of the resulting systems [27].

A set of business rules is an expression of business logic in a special-purpose programming language. The business rule interpreter should be well-enough defined and well-enough implemented such that a ruleset and its interpreter will always produce the same business results despite changes made to other aspects of the software and hardware

platform. Microsoft's BizTalk is a prominent example of this approach [2]. There are many others, both in open-standard proposals and in proprietary languages. Most business rule systems are aimed primarily at the proper implementation of a set of business rules, rather than at the verification or proper design of the ruleset as a whole [2]. IBM's Business Rules Markup Language is a counterexample: it was designed to have clean semantics, which are amenable to analysis by defeasible reasoning [33]. However it has not found much acceptance in the business market, perhaps because few programmers or analysts are willing to use a logic programming language to express their business rules [2].

1.6 Organization of the Thesis

In Chapter 1 we introduce some basic background knowledge on risk analysis in B2B e-commerce, and give out our purpose for this research of security analysis on business logic level. The next chapter concerns the fairness as an important aspect on the business logic and the concepts defined for describing it. Also, we introduce briefly the AARN payment system and announce the constraints of our case study. Then, in Chapter 3 we search the formal analysis modelings and discuss them. We explain why we chose form-oriented analysis as our method for modeling and analyzing the security.

Chapter 4 concerns the application of DTIM and UMM in the modeling. Here we determine the formula for fairness and use it to check the states of the parties. Finally we present the result and argue that the recourse process influences the records between TIP and the vendor. In Chapter 5 the same method is applied to a new design. The analysis locates two risks in the new design, and present the solution for the risks. Finally, Chapter 6 contains the summary of the results and achievement of this thesis and move forward to future research in applying form-oriented analysis method in secure modeling and developing practical form-oriented analysis tools.

2
B2B in E-commerce

The related e-commerce concepts of fair exchange and fairness, trustworthiness and trust form the basis of the application of the B2B payment system. Based on these concepts, we explore the theoretical issues about security in e-commerce. The application asks for security requirements, which we will analyze by our new modeling method in the later chapters.

2.1 Fairness and Fair Exchange

In E-commerce systems, an aspect of business logic is to promise fairness. As the feature ensuring parties conduct their business to their mutual moral standards, fairness is one of the paramount features for e-commerce payment systems. Originally business people set up their business as a fair one to attract customers. However, in many previous research, keeping the feature of fairness was not a important security issue.

2.1.1 Security is more than a technical issue

Security is a technical issue, with important consequence for business. We can categorize the concept of security in e-commerce by the goals. First of all, it has to ensure the

whole system is running correct, which we call its reliability feature for the system. For example, when there are user inputs, the system has to do some data checking assignments. After the system has reliable security, we can acquire more extended features, such as to prevent the frauds or invasions during data transfer communication, which we call the safety feature.

As the paramount consideration for security process, reliability ensure the correctness of the execution of the process. In a reliable system, the business will run as expected, and the actions of the business parties are predictable and responsible. The behaviors of the parties are expected, harmful or harmless. In the situation of expected harmful behaviors, the system should either have the solution to deny or execute the behaviors and approach reasonable results. If the system is not reliable, the results of the execution could be unpredictable and extremely strange. Every behavior can lead to a wrong result as no trust between the business parties will exist.

On the other hand, safety of the security e-commerce system concerns protecting the privacy and the data, including prevention of fraud and invasion. Safety in e-commerce covers all of the computer security issues, e.g confidentiality, integrity, and availability.

Regarding e-commerce payment issues, the goal of security concerns not only preventing the system from malicious parties' faking or attacking the transfer communication, but also ensuring the correct execution of the transaction. Here the correct execution means the parties of the e-commerce should behave according to their promise, e.g. their legal contract. As we mentioned, the correct execution is a feature of reliability. However, as the parties in e-commerce transact in an invisible environment, a **fraud** party may cheat and benefit from the other parties' honest behavior, which is not fair to the honest parties. Therefore we can conclude that fairness is very important for reliability of the security. This feature helps to protect the business.

2.1.2 Fairness

When people talk about security of a business, they focus on the safety of the transfer process, correctness of the number of items, or the authentication of the business parties, and often ignore a basic rule that the parties should behave as they said. To avoid behaviors of either business party that would aggrieve the other, the commerce must involve the concept of fairness. Fairness answers the issues of misbehavior, not behaving or overbehavior of the parties.

As a basic rule of transaction, fairness is an essential issue for commerce. In a business activity, a goods seller exchanges his goods with a buyer for money, which in this thesis will be called an exchange between the business parties. The customers are both expecting their acquisition is worth what they paid. That is, as long as the exchange deals, what they gain is equal to what they paid. They might continually purchase what they gain at a higher value, which enables others to earn more profit, as long as they can find that new party and make the transaction. But that will be another exchange and in that exchange both of the parties will expect the same. In addition, in an exchange the parties expect the result will be either successful, which means both parties receive what they want, or canceled, which means both parties take back what they have without losing anything.

We follow the definition of fair exchange in [32]. "Assuming two parties A and B, each of them starts with an item i and a description d of what that party would like to receive in exchange for i." The notation identify the items using subscripts, i.e. i_A is As item and d_B is the related description of Bs desired item. Assume there exists a verification function which takes an item and a description and returns the value *true* only if the item matches the description. There are two possible termination states of the protocol, either *success* or *cancel*. Both parties need to check whether and in which state the protocol has terminated. Therefore we have a fair exchange protocol, if it implements these requirements [3, 32].

Effectiveness is achieved if both parties behave according to the protocol, and both of them do not want to cancel the exchange, and both items match the description, then when

the protocol is completed, A has i_B and B has i_A and both reach a *success* termination state. Termination occurs when a party who behaves according to the protocol will eventually reach either a *success* or a *cancel* termination state. The last in this string of definitions is a preliminary definition of fairness: At least one party does not behave according to the protocol or at least one item does not match the description, but no honest participant wins or lose anything valuable, fairness has occurred.

As we can see, only when both of the parties check the terminated states can the fairness of the exchange be verified. It is a strict verification for an exchange, as it can protect the benefits of both parties. This strict fairness requirement for both parties is called strong fairness [3, 32]. If the exchange process is long enough to separate the verification of the parties, we have a weak fairness termination. It describes the situation when the parties can only prove themselves if they behave correctly, such as when the exchange verifications are asynchronized. For example, in a weak fairness termination, if party A proved they followed the prescribed protocol, then it provided a fair exchange to party B. But if B at this time can not prove their behaviors are correct, it is still accepted as a weak fairness termination until later on party A proved party B's behaviors were incorrect.

A fair exchange is still termination-related. Fairness can not promise the exchange be successful; it promises that even when the exchange failing, all the parties can get their items back. A successful fair exchange is terminated at a *success* state, that is, both parties receive the items from the other party. A failing fair exchange is terminated at a *cancel* state, that is, both parties do not receive the items from each other.

The tough thing in the fairness verification is who will be in charge of judging the proof; in most cases a court and a lawsuit are involved. However, it is unclear how acceptable the proof is, and lawsuits are always expensive with an unpredictable outcome. To avoid the complexity and expense of the lawsuit, either of the business parties may try to keep the exchange in a zone of strong fairness. As Henning Pagnia et al have proved in [32], a fair exchange has to involve a third party to judge the fitfulness of the exchange to the fairness.

A third party working in the exchange transaction should be accepted by both of the business parties. In other words, a third party should be trusted by both of the business parties. There is a semi-trusted third party issue, which means the third party may misbehave themselves, but does not conspire with either of the business parties. According to the requirements from the payment system, in our B2B e-commerce model, we are looking for correct behaving third party, so we won't discuss the semi-trusted third party in this thesis.

2.2 Trust in E-commerce

In fair exchange, a trusted third party can insure the exchange runs in an expected way. Trust is usually considered as attitude and disposition of belief in competence, integrity and predictability [23, 47]. If a member believes his partner has the capability to execute his request and the willingness to do that, or if the member believes that his partner can be relied on and his behaviors are consistent enough, or if a member believes that the business will keep on going transparently to him and under his control, willingness or expectation, we say the member trusts his partner [23]. The trustor either believe the trustee, or he can predict the trustee's behavior, or he can control the trustee. If the trustee is worth enough for the trustor to take the risks of unbelievable, unpredictable or uncontrollable things happening, and chooses to trust him, the turstee is trustworthy. In practical commerce, trustworthiness usually gains from long term cooperation, good business credits and formal legal contracts.

Business parties need to trust each other because they believe that trust can lower their business risk [36]. In traditional commerce, if the parties trust each other enough, it is not necessary that the exchange transaction always be fair; they can take the risk that either party was assaulted in one exchange, to keep the business relationship alive for further benefits in a long term. However, this does not work in e-commerce. In e-commerce, the business parties may not be familiar with each other, or even not heard each other before. In an e-commerce business, what the members are facing are all the

signals on the computer network instead of the real, physical entity. It is quite hard for a member to feel that his partner is stable or trusted on the first deal. After all, any information from the web could be faked [43]. It will extremely lower the effectiveness of the business for the parties to run through the steps of checking and building trust, if it is not impossible; it is much harder for people to check the abstract electronic signals than the physical entities in every single business [43].

As a solution to building the trust more easily and lowering the risks, a model of the trusted third party has been developed in the B2B e-commerce [42]. In this model, the business parties exchange and deal through a third party which trust both of the parties and is trusted by both of them as well. They trust the third party rather than each other. They have built a stable and consistent relationship with the third partner individually and can exchange through the trusted third party. The third party can support the trust for the parties by the conventional ways already mentioned. After the parties make the deal, the third party will look after the business, such as goods delivery, bill payment and banking transfers business for the parties. As the third party trust both of the parties, it guarantees the payment process to both of them. In this paper, we call this third party as a trusted intermediate party (TIP).

Another important benefit for the parties is that the e-commerce service from a third party can lower the high cost of development which is making proprietary or custom hardware and software unattractive [48]. Therefore, more and more small business organizations are coming to buy a TIP service as their e-commerce solution. As the TIP is acting in such a key role in the B2B e-commerce, we need to be concerned more with the security of the TIP and the secure processing of its business logic.

2.3 AARN B2B Payment System

As a B2B payment service provider, AARN Innovation Limited provides an intermediate e-commerce service to the business parties to reduce their cost on reconciliation. This is the AARN payment system which used to be called as EDIS i-payment system. It has

established a network for hundreds of small or medium-size companies and organizations in New Zealand. These businesses have developed trust with AARN and have authenticated AARN to manage their payment process from their bank account. As each of the parties has its own bank account, it is very hard for the system to connect with each bank and get the authority to transfer funding from the business parties account. The engineers of AARN decided to build an architecture based on the TIP theory. In the architecture, we will band the payment system to a unique bank as the intermediate bank; all the transfer processes with the business parties are either started from this intermediate bank, or terminated at the bank. The reason we involve an intermediate bank is, the internal transfer processes between banks are mature and stable. Banks have found the way to do this business and built the standard for transferring funds through different banks, even oversea banks.

For our puropses, a business role is an organization which is willing to do some business with other organizations. There are several kinds of business roles involved in our B2B service, the vendor(s), the customer(s), the TIP and their banks. A vendor is a party who is going to exchange his goods for some money. A customer is a party who is going to exchange money for some goods he needs. A vendor and a customer are both called business parties. The goods from the vendor and the money from the customer are their items respectively. The TIP acts according to the definition in 2.2.

One transfer service includes three processes: the process of contract, the process of delivery, and the process of payment. As we are only concerned with the payment process, we describe only that process. A payment process includes many transactions. A transaction is a set of behaviors by one or two parties, to provide a function, such as storing data or removing records. All these functions together, provide the funding transfer function of the TIP. To approach the transfer function, business provides two kinds of payment method, Direct Debit payment (D/D) and Direct Credit payment (D/C). In a D/D payment, the payee is authored to access a nominated customer account by the agreement they made. The payee debits the payment directly from this nominated account to his own account. If the payment proceeding from the observed way, we have a D/C

payment. In a D/C payment, the customer deposits the payment to the payee's account.

In addition, to ensure benefit to the customer in a D/D payment and to predict the abuse of the D/D payment by the payee, between the financial organizations whose financial service can help complete the payment, an agreement is in place where they provide a recourse transaction after the D/D payment. Within a valid period, if the customer applies for the recourse transaction, the financial organization forces the payment back without any limitation from the payee. This valid period in New Zealand is 2 days.

Besides these definitions, in our case study, we set up the business using this business logic: first, the vendor delivers his goods or service to the customer; after both of them identify the delivery, the customer pays the vendor.

The initial purpose of our research was to help AARN build a formal model to effectively analyze the potential operational risks of the system. In this research, we are concerned with the requirements on the business logic level such as recourse transaction. By applying a form-based analysis method (see section 3.5), we proved that some requirements can partially conflict with the security requirement for the system. Meanwhile, we used the form-based analysis with the security analysis area. It is fairly new for form-based analysis method to be applied in this area.

2.4 Problem Setting

In our case, AARN works as a TIP but as we discussed in 2.3 it is still not a formal strong fair exchange model, nor a weak one. In this case, the TIP can not promise the case will a strong fair exchange, because it can not control anything of the delivery process. The TIP can only work after the customer receives the items. By the time the payment starts, the vendor has already provided its fair exchange to the customer. Therefore, we are concerned only with the process of the payment transfer. The fairness of the transfer process needs to be strong, because we have to make sure the customer also provides its fair exchange. We define this as a semi-fair exchange, but it still promises fairness during the payment transfer process, and protects all the parties' benefit only during the

payment process. Any problem in the delivery process should be argued by the business parties, or involved in an authority organization as the trusted third party.

On the other hand, as a service provider, AARN would like the whole system to work as safely as it can. Therefore, if the payment process of the fair exchange is particularly strong, the risk of the TIP will be lowest. Although the risks of the business parties are also limited, the service provider concern the TIP as a primary risk.

Additionally, service providers face the problem of illustrating the features and benefits of their system to their potential customers – the business parties. As a requirement from the market, the easiest diagrams and contents are the best way to illustrate the work processes of the system to a customer who has no technical background but lots of interest in how the system works.

Banks must consider both financial risk and operational risk in all their operations [5]. An operational risk is the risk of loss due to inadequate or failed internal processes, or systems, human error or practice, or from external events [5]. A financial risk, on the other hand, involves capital flow, including profit risk and liquidity risk. Vendors, customers, and TIPs categorize their risks as either financial or operational ones.

As a business level analysis, our research greatly concerns financial risks. Also because our case study studies the fairness of the payment, liquidity risk is not in our analysis. We will focus only on profit risk. No business owner wants to risk losing money when engaging in any transaction. The operational risk in the modeled e-payment system is at an implementation level, below our level of analysis. Such risk is already fairly well managed by well-known techniques in information security. For example, using cryptographic hashes to ensure message integrity would make it very difficult for an adversary to falsify a message. Using robust message-passing protocols, it is very rare for a message to become "lost" or be misdelivered.

Therefore our purpose of the modeling is to formalize the case in a simple and understandable modeling so we approach a strong fair exchange model for the payment processes. We expect our modeling can detect complex and serious fraud risks in the business, but in our research we only focus on a simple case and try to detect the fraud

in this case. We assume the items have been accepted by the customer and the vendor is waiting for the pay back. To transfer funding from the customer to the vendor, the TIP should have the authority identified by the business parties on their bank accounts. Also, the profits or benefits of the TIP are not included in this modeling, as it only concentrates on the roles in the transfer processes.

2.5 Summary

In this chapter, we discussed a specified business environment, the fair exchange. In a fair exchange, the business logic requires the business parties examine their terminal states. Also we described the business logic for a fair exchange. In our case study, we are trying to create a model for the real payment system, which fits the logic requirements of the fair exchange. The concepts we defined in this chapter will direct the formal modeling in the following chapters.

3
Language for Modeling Our Systems

To clearly describe the problems existing in the systems, we need to develop modeling language. In the business logic analysis, we need the modeling language to be a strong linguistic tool that presents meanings and descriptions of the business. In this chapter, we introduce some modeling languages and methods which might fit our purposes for the system.

3.1 The Evaluation of Modeling Language

Modeling languages, according to the definition from the Wiki Encyclopedia [46], enable researchers to specify and clarify the requirements of an organizational system on the business level. These languages are used to visually explain the system requirements and clarify the functional specification and prototype in a way that management, user groups, or other stake holders can understand, with a goal of eliciting feedback from these groups. Several modeling languages are used in security related modeling, such as UML, Communicating Sequential Processes (CSP), and Petri Nets. Every modeling language has its own specification and syntax, to approach a specific purpose and solve one kind of problem. As in business logic analysis, we should choose a proper modeling language

for the case we study.

Our target of the case study is to find a formal modeling language that can:

- Describe the business in a simple and comprehensive model with limited symbols and constraints, so that our clients can easily understand and confirm our analysis.
- Easily extend the semantics and syntax while keeping the constraints with the existed formal methods.
- Be used in analyzing business logic, focusing on abstract functions for the security issues of profit risk, and proving fairness.

We explored UML, CSP and Petri Nets for this purpose, and briefly discuss them in the next few sections.

3.2 Unified Modeling Language

UML is the most popular modeling language for computer architecture, design, and analysis. Because of its object-oriented features, it has been successfully used in many applications [10, 13].

As applied in many areas, UML can also be used for security purposes by extending specific characteristics. Within a formalized design model, UML can work for automatic verification [17]. Currently there are couple of extensions of UML used for security purposes.

SecureUML is an example of a security application of UML. By applying different patterns related to access control infrastructures, it extends UML to support constraints of authorization [17]. "SecureUML defines a set of stereotypes and uses them to automatically produce the code that controls how users can access the different parts of the application. These stereotypes consider only the static parts of UML; dynamic ones are part of the future work" [17]. SecureUML integrates access control information and defines the security system by models and is constructed by these models [27]. As an

application of UML, SecureUML describes the business based on objects and does not clearly define the process steps of the business logic. In the example of [27], it separates the roles (e.g user), for the access control, but it does not define what is a proper process for controlling a role. Instead, the objects in SecureUML are the majority on presenting the security.

Another ambitious UML profile is UMLsec [17], which extends state charts and component diagrams on the following applications: fair exchange, confidentiality, secure information flow, and secure communication link [20]. UMLsec extends formal semantics based on Abstract State Machines (ASM) language to fit the analysis of model-driven design. The purpose of UMLsec includes but is not limited to the following:

- Evaluating the vulnerabilities in the model for the specifications of UML;
- Encapsulating established rules of security engineering into the model;
- Providing a general way for developers to check UML models in security;
- Considering security from the application design level;
- Effectively and economically modeling verification [17].

In UMLsec, common security requirements are designed as stereotypes with tags (secrecy, integrity, etc) [17]. UMLsec evaluates specifications and indicates possible vulnerabilities by applying associated constraints [17]. It describes the static security requirements by giving security policy and the specification by which UML satisfies the defined requirements [17]. As to the enforced requirements, such as secrecy and integrity, UMLsec provides basic security stereotypes, and it allows analysis of different threat scenarios according to the strengths of the attacker. Additionally, it allows integration of security concepts such as access control and user authentication into general applications as components, and it provides simple security features such as symmetric encryption. The main goal of UMLsec is to "automatically check security properties on an extended UML model, stored in XMI [abbreviation for XML Metadata Interchange] standard format so that a

formal tool can detect the vulnerabilities in the model to verify security requirements" [17].

UMLsec describes business logic better than SecureUML because it also covers the work flow and the security issues in the work flow. But there is no analysis method nor related constraints to enhance its capability for description. It constructs the implementation as classes and can be used to define the function for the classes. But the way it describes the relation to the objects and the message exchanging is complex and messy, which is hard for communication and explanation to clients.

3.2.1 Object Constraint Language

Object Constraint Language (OCL) is an associated constraint language in UML. OCL provides a formal description of a framework for specifying constraints on a model; specifically, it is designed for presenting additional constraint information. It can provide a special set of annotations that impose additional restrictions on a model as the constraints [34]. As a textual constraint language, OCL defines declarative expressions which are also free of side effects. In addition, OCL supports the notational style similar to common object-oriented languages. Model designers can define specific constraints on an abstract level as simple and clear concepts to help describe adequate information from lower level implementation details.

As a textual language that allows specification of additional constraints, OCL usually is combined with a UML model to enhance the language quality of description. Usually a model itself provides the context for constraints as the description of the application [34]. As an example, let us assume a requirement in a UML diagram is that "all department objects can not have a negative budget" [34]. As a data constraint of the UML, we have

$context$ Department inv :

$self.budget >= 0$ [34].

By using OCL in UML diagrams, we add more information into the diagrams, which helps the model describe things more correctly, especially in the security modeling. In

fact, OCL can help UMLsec on constraints.

3.3 Communicating Sequential Processes

Communicating Sequential Processes, or CSP, is "a simple programming language designed for multiprocessor machines" [19]. It was developed based on the assumption that every process has its own processor to execute on and only on a message passing through the concurrent processes as the solution of communications between these processes [19]. The key feature of CSP is "its exclusive reliance on non-buffered message passing with explicit naming of source and destination processes" [19]. For simulating the concurrent processes, CSP uses guarded commands inside alternative commands to allow a process to wait for messages coming from different sources [19].

CSP is applied in security areas, especially in parallel command execution [8], non-interference between processes, and access control [9]. As its features have shown, CSP describes the processes which can perform events. CSP conventions use the typological conventions of upper case letters for processes, lower case letters for events, and a capital initial letter for names of sets of events [9]. The ultimate unit in the behavior of a process is an event. Events are regarded as instantaneous; if we wish to represent an activity with duration, we must introduce two events to represent its start and finish so that other events can occur in between [8]. CSP is "an abstract language designed specifically for the description of communication patterns of concurrent systems components that interact through message passing" [39]. Researchers have applied CSP to the analysis of non-repudiation protocol for a weak fair exchange [28]. Here CSP is mainly involved in describing the processes dealing with encryption and authentication of the receipt or stub. Although it functions to describe the processes, it hardly has a clear abstract description of processes in a specific application. The semantics of the simple symbols can not fully express the requirement of the application, for example a key meaning message for a specific commerce requirement like a D/D payment requirement.

CSP provides a very formal and precise way to describe the behavior of a collection

of message-passing processes [40, 35]. Many properties of these processes are suitable for analyzing, due to the restrictions in the language. However the very formality and precision of this language seems to make it unattractive to most designers, and it is certainly almost incomprehensible to anyone without a strong technical background in computer science [2]. The proof process is strongly related to mathematic semantics, which might be a challenge to the clients. All these reasons makes it unsuitable to be our target modeling method.

3.4 Petri Nets

A Petri net is a very nice formalism for expressing token-passing systems [21]. It has been extended to the modeling of message-passing systems, in several different ways.

In a very initial research paper [30], Murata gave out most of the initial concepts and crucial constraints to which we refer as an illustration: As a specific kind of double-directed graph, it contains a start state called the initial marking, or MO. The underlying graph N of a Petri net is a directed, weighted, bipartite graph consisting of two kinds of nodes, called places and transitions, where arcs are either from a place to a transition or from a transition to a place. In graphical representation, places are drawn as circles, transitions as bars or boxes. Arcs are labeled with their weights (positive integers), where a k-weighted arc can be interpreted as the set of k parallel arcs. Labels for unity weight are usually omitted. A marking (state) assigns to each place a nonnegative integer. If a marking assigns to place p a nonnegative integer k, we say that p is marked with k tokens. Pictorially, we place k black dots (tokens) in place p. A marking is denoted by M, an m-vector, where m is the total number of places. The pth component of M, denoted by $M(p)$, is the number of tokens in place p. In modeling, using the concept of conditions and events, places represent conditions, and transitions represent events. A transition (an event) has a certain number of input and output places representing the pre-conditions and post conditions of the event, respectively. The presence of a token in a place is interpreted as holding the truth of the condition associated with the place.

In another interpretation, k tokens are put in a place to indicate that k data items or resources are available. There are some typical interpretations of transitions and their input places and output places in [30]. A formal definition of a Petri net is also given in [30]. An additional description is, "A Petri net is a directed bipartite graph with nodes called places, depicted as circles, and transitions, depicted as bars. The edges of the graph are called arcs. A marking is an assignment of an integer to each place in the net that represents the number of tokens, depicted as black dots, at that place" [16].

From the previous description, we can have the impression of Petri nets that it is a state-based message-passing modeling tool. However, to express the business logic, the constraints of the Petri nets are somehow weak, and its formal states descriptions can not contain more information above the directions. Furthermore, its mathematical structure is strict but not flexible enough. The analysis with Petri Nets is based on the status of each point, and the result set of the analysis can have no relationship with the diagram. Besides, as the number of points increases, the diagrams can be complex, but the symbols of the diagrams still only describe the business abstractly.

3.5 Form-oriented Analysis

From our discussion in the previous sections, among the former modeling languages and tools, UMLsec probably come closest to being our target modeling method. However, as an abstract architect modeling tool, it is still too rough to describe all the required information. Some important information such as the sequence of the processes must be put in the comment. Meanwhile, it is hard to connect the process with the program design; the automatic checking process can still not transfer it into the detail program design level, such as activity diagrams or state diagrams. In addition, it defines tags of start and stop, but the sequence diagram can not properly and clearly describe the whole application in considering the objects and roles. Therefore we choose to use a simpler but more abstract and compact modeling language for our simple business logic description, and we find form-oriented analysis.

Form-oriented analysis approaches to the modeling of certain general issues with a *submit/response* interaction paradigm [14]. Submit/response style applications are the applications wherein the user exchanges information by a bundle of data; then the system replies with the result of the data execution. In addition, the reply may include a further step for the information exchange. Originally, form-oriented analysis worked for well-known form-based applications ranging from typical internet shops through supply chain management to flight reservation systems [14]. However, we can apply this modeling in a model of interaction between different actors, our so-called parties.

3.5.1 DTIM

Form-oriented analysis can be applied to different models. In this paper we discuss the application for modeling communications between automated systems, which in our case are the communications between the payment system, the business parties' Enterprise Resource Planning system (ERP) and the banking transfer system. The communications are generally based on messages. The form-oriented analysis method uses Data Type Interchange Models (DTIM) to represent the involved types of messages and systems [15].

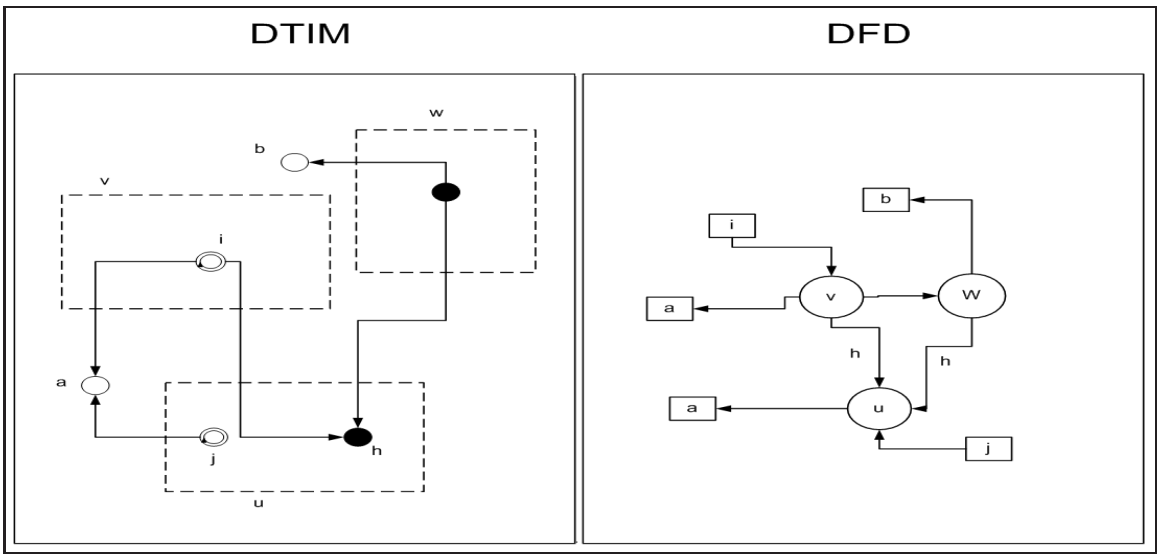

Figure 3.1: A DTIM and its equivalent DFD.

A DTIM is similar to a Data Flow Diagram (DFD). DFD is usually used to explain the basic logic of the application. It is simple and convenient, but has less information,

especially no timing. DTIM can reconstruct DFD into a formal model. As an example, Figure 3.1 illustrates a DTIM and its equivalent DFD which describe the same business logic respectively.

A wonderful feature of DTIM is that it allows the extension on diagram signals. As a business logic modeling method, DTIM does not limit its semantics. By adding customer-defined signals, it can express specific business logic under the special requirement. Also, we can compose some associate modeling language on DTIM, such as UMM.

3.5.2 UMM

UMM is a strict subset of OCL 3.2.1. It describes only messages exchanged between the parties. The messages are usually determinative in a specific system, so that a type, or we call it a class, can adequately describe the message. As UMM is a class model, the instances of the model form strict hierarchical structures, and they can build complex messages by the composition of the UMM associations [15].

In form-oriented analysis, UMM has two parts – the UMM data structure and the business constraints. The data structure defines the types of the messages in the transactions. The business constraints define the messages exchanged between the transactions. In a business constraint, there is a source transaction and a target transaction. The output specification expresses constraints to the message that is output to the target. A "flow condition" expresses the condition under which the message is sent; it is usually a "select" condition or "if" condition. The business constraints also have methods to support business behavior. For example, $target.tp.add(source.tx)$ means the message tx from the source transaction will be added into the message tp of the target transaction.

To illustrate the work for DTIM and UMM, we use the following scenario.

There is a very simple search engine for the library online system with two parts of data for the search function, the local database and the remote database. Here the local database is the storage for the books and documents in the library, and the remote database is a larger online storage which provides the library service. To make the example

simple, we assume the search cover only the title of a book. The business logic of this scenario is in the following steps:

1) The system starts searching by a name of title, and deals with the title including trimming it and analyzing the literature, and sends the searching key words to the database search engine.

2) The database search engine looks through the data in local database and replies with the result(s).

3) If the engine found the book(s) named by the querying title, it lists its(their) bibliography with abstracts.

4) After the list, the system also provides an extending search on the remote database. It sends the searching key words to the remote database.

5) The remote search engine looks through the data in the remote database.

6) The remote search engine collect all the results from the remote database, and sends them back to the local.

7) The system lists all the results from remote database.

The diagram in Figure 3.2 illustrates the process and its business logic. In this DTIM diagram, we restrict several kinds of symbols to different kind of descriptions. For these descriptions we define different types of circles shown in Figure 3.2: a **double-circle** is a starting point for a transfer process; a **bold circle** is an ending point; and a regular-width circle is a mid-point in a transaction (every system diagram can have many regular-width circles).

To extend the formal description of this diagram, we define the data structure of the messages transfer between the transactions as follows:

```
Transaction > A
    title: String 1..1
Transaction > B
    title: String 1..1
Transaction > C
    title: String 1..1
Transaction > D
    title: String 1..1
    list: bibliography 0..n
```

```
Transaction > E
    title: String 1..1
Transaction > F
    title: String 1..1
    list: bibliography 0..n
Transaction > G
    title: String 1..1
    list: bibliography 0..n
```

Here in this UMM data structure, transaction A is the start transaction that the search engine receives the title of a book from the client. Transaction B is searching the books in local database and sends the title to transactions C and D. Transaction C starts a search on the remote database and obtains the title from transaction B, then sends it to the remote. Transaction D receives the search result – a bibliography list. Transaction E receives the title through the network, starts the search, and sends it to transaction F. Transaction F then receives the search result (another bibliography list) and sends both the title and the result back to the local. Lastly, transaction G receives the title and the bibliography list and presents them.

In Transaction A, there is one message, the title of the aiming book. Similarly, in

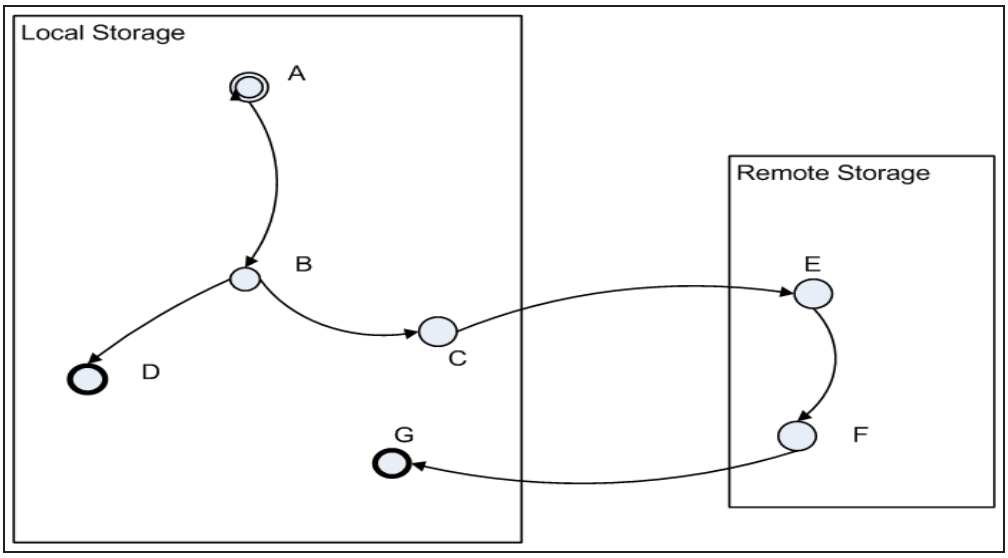

Figure 3.2: An example for a book search engine.

Transactions B, C, E we also need the title message. On the other hand, in Transactions D, F, G, a bibliography list is also needed.

Therefore, according to the business logic, the UMM constraints for the DTIM diagram are:

1) A to B: Output: $target.title = source.title$

// this equation expresses the message exchanges between the transactions.

//An output explains the direction of the exchange and the content of the message.

2) B to C: Output: $target.title = source.title$

//the search engine looks through all the data in the local storage

B to D: Output: $target.title = source.title$ Flow: $source.title = B \rightarrow allinstances() \rightarrow select(title).title$

//A flow expresses a selection or a searching behavior.

//$select(title).title$ expresses selecting the list by the condition of title.

Output: $target.list.add(source.title)$

//This output adds the result records into a list.

3) C to E: Output: $target.title = source.title$ //the search engine looks through all the data in the remote storage

4) E to F: Output: $target.title = source.title$

Flow: $source.total = B \rightarrow allinstances() \rightarrow select(title).title$

Output: $target.list.add(source.title)$

5) F to G: Output: $target.title = source.title$

$target.list = source.list$

From the DTIM diagram and UMM constraints above we can see, as a formal description method, form-oriented analysis works well to describe the business logic of the example as formal modeling. This formal modeling for the business logic can be used for the basic discussion for the application from business logic level.

3.6 Summary

We can see this feature of modeling language description: there is no unique answer for the modeling language to represent the real questions. However, there are limitations of the requirements which can influence the performance of the modeling langauge. Even for the same business question, we choose different modeling languages or combinations of them under different requirements. As an example, in the case of our B2B fair exchange modeling, we are more focused on the business logic content description and its external information. Meanwhile, considering the requirement of formal language design, we choose to apply the form-oriented modeling for the case study. However, this does not mean all the other languages are not fit for the model. In addition, many of the modeling languages are improving the semantics and symbols to extend the application area. Therefore, there maybe other modeling languages suitable for this case study, but we just choose the best one we could see. Form-oriented analysis modeling seems to be a promising and novel approach to model the business logic of applications. We choose it as the method of modeling the business logic of the AARN payment system in the next chapter.

4
Modeling the AARN Payment System

In this chapter we discuss the modeling of our case, the AARN payment system. We illustrate how to use the DTIM as the measure of form-based analysis for the formal presentation of the structure of the B2B e-payment system using a trusted intermediate party (TIP). There are five parties in this system: a vendor, a customer, the TIP, customer bank *bankC*, and vendor bank *bankV*.. First we setup some specific constraints for our DTIM diagrams, then we model the current existing payment system and analyze the model.

4.1 The Constraints

A DTIM diagram can represent messages, types of messages, and the relationship between parties and the subsystem in the whole system. Despite the simplicity of the basic DTIM diagram symbols, it allows users to extend their own symbol types which have to be compatible with the basic symbols as a sub-class with specific meaning, as we mentioned in 3.5.1. In our case, based on the definition of fair exchange, the semantics of our DTIM diagram must include a "start" point and an "end" point. We use the definitions for the double-circle, bold circle and regular-width circle, mentioned in Chapter 3. For additional

expressiveness we define different types of symbols shown in Figure 4.1: labels on nodes indicate message types as identifiers. We require that all messages in a DTIM have different nodes so that DTIMs have similar semantics to an object-oriented programming language, where a DTIM system type is analogous to a class and a DTIM message is analogous to a method call. Multiple instances of a single system are possible, but in this thesis we will analyze one instance only for each system type.

Dotted lines in our DTIMs connect comment boxes to circles, as seen in the following Figure 4.2. On Step 6, BankC sends either a "Cancel Payment" message to the TIP, or a "success" message to BankV. We also use dotted lines to connect comment boxes to arcs, or to pairs of arcs, to indicate special semantics as with the "XOR" messaging in Steps 4 and 6. Furthermore we use ranges, such as the "0..1" near the in-arc of transaction I in the TIP, to indicate a variable number of outgoing messages represented by an arc. In the case of node I, it may not be invoked at all or only once. Finally, we use a straight line to express the relation of proviso between two independent transactions in a time-based consequence. For example, in Figure 4.3, after transaction E happens, the transaction L may happen (or not; we use the range "0..1" to express this) later on, but L is not directed by E. In fact, L is another *start* transaction for a parallel process.

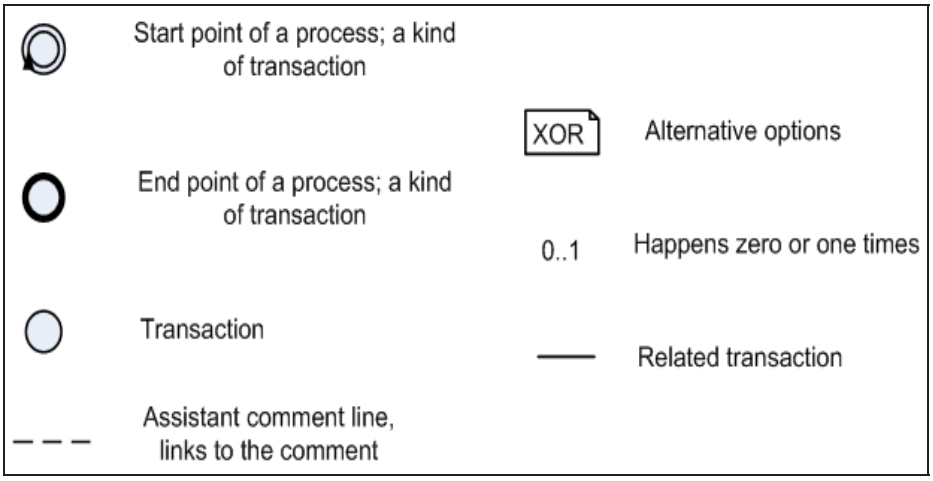

Figure 4.1: Custom signals.

4.2 The Payment Transfer Architecture

As the computer architecture for a business, the model for the payment transfer has to be reliable and secure. As the transaction will be either successful transfer or failing possibilities, the concern is to prove the transfer is a fair exchange to avoid profit risk. Also, the model should provide evidence for the security issue that there is no other unexpected events in the transfer. In our case, we model the architecture and re-construct it out of concern for events of recourse transaction. In the first architecture, we are not modeling the intermediate bank we mentioned in 2.3. The discussion about the intermediate bank will be on the next chapter.

4.2.1 The Simple Economic Model

The AARN payment system concerns transferring payment between business parties. A payment transfer process involves a vendor, a customer, their banks, and a TIP. This simple scenario illustrates the process: the payment transfer process starts after the customer receives the items from the vendor. The TIP first receives an invoice of sale from the vendor and then asks the customer to approve the invoice. The customer accepts the invoice and replies to the TIP with confirming the TIP contacts bankC to request sending funds to bankV. The scenario describes the fundamental steps of the payment process. According to the scenario, we describe the transfer steps as following business logic:

- **Step 1** The vendor sends an invoice to the TIP, [GOTO **Step 2**];
- **Step 2** The TIP forwards the invoice to the customer and records it as a log record, [GOTO **Step 3**];
- **Step 3** The customer accepts the invoice, and sends out a payment instruction to the TIP, [GOTO **Step 4**];
- **Step 4** The TIP records the payment instruction and checks it with the invoice record; if they match, the TIP will send payment instructions to bankC, [GOTO **Step 6**]; otherwise it will mark the payment as failed, [GOTO **Step 5**];

- **Step 5** The TIP rejects the failed payment instruction and notifies both parties [STOP];
- **Step 6** BankC transfers the funds to bankV on the order of the TIP; if the transfer is successful, the process is finished [STOP]; otherwise, bankC will send a failure notice to the AARN payment system [GOTO **Step 7**];
- **Step 7** The TIP sends a failure notice to both of the parties [STOP].

This business logic describes a business transfer process in a simple situation. Our research is based on this simple economic model and explores the security under different possible situations.

4.2.2 The Payment Transfer Architecture

In a form oriented analysis, we say that the types of the messages sent between the systems are collected in a UMM that complements the DTIM diagram. In our case, the UMM data structure looks as follows:

```
Transaction > A
    total: Integer 1..1
    invoiceID: invoice 1..1
Transaction > B
    total: Integer 1..1
    invoiceID: invoice 1..1
Transaction > C
    total: Integer 1..1
    paymentID: payment 1..1
Transaction > D
    total: Integer 1..1
    invoiceID: invoice 1..1
    paymentID: payment 1..1
    bankAccountID1: 1..1
    bankAccountID2: 1..1
```

```
Transaction > G
    total: Integer 1..1
    invoiceID: invoice 1..1
    paymentID: payment 1..1
Transaction > H1
    total: Integer 1..1
    paymentID
Transaction > H2
    total: Integer 1..1
    invoiceID: invoice 1..1
Transaction > I
    total: Integer 1..1
    invoiceID: invoice 1..1
    paymentID: payment 1..1
```

```
Transaction > E
    total: Integer 1..1
    bankAccountID1: 1..1
    bankAccountID2: 1..1
Transaction > F
```

```
Transaction > K1
    total: Integer 1..1
    paymentID: payment 1..1
Transaction > K2
    total: Integer 1..1
    invoiceID: invoice 1..1
```

According to the business logic in 4.2.1, we setup the UMM constraints for this architecture as:

Step 1 A to B: Output: $target.total = source.total$ //the amount of money from the invoice recorded into the sytem
$target.invoiceID = source.invoiceID$

Step 2 B to C: Output: $target.total = source.total$ //the amount of the invoice equals the money the buyer will pay

Step 3 C to D: Output: $target.total = source.total$
$target.paymentID = source.paymentID$

Step 4 D to E: Flow: $source.total = B \rightarrow allinstances() \rightarrow select(invoiceID).total$
Output: $target.total = source.total$
$target.bankAccountID1 = source.bankAccountID1$
$target.bankAccountID2 = source.bankAccountID2$
D to I: Flow: else Output: Fail message

Step 5 I to K1: Output: $target.invoiceID = source.invoiceID$
message(fail)
I to K2: Output: $target.paymentID = source.paymentID$
message(fail)

Step 6 E to F: Success message
E to G: Output: $target.total = source.total$

Step 7 G to H1: Output: $target.invoiceID = source.invoiceID$

message(fail)

G to H2: Output: $target.paymentID = source.paymentID$

message(fail)

A diagram of the payment transfer architecture in the DTIM diagram is shown in Figure 4.2. As an extended feature, all the transactions are automatically logged as records by this system. This feature relates to the details such as the implementation of the system.

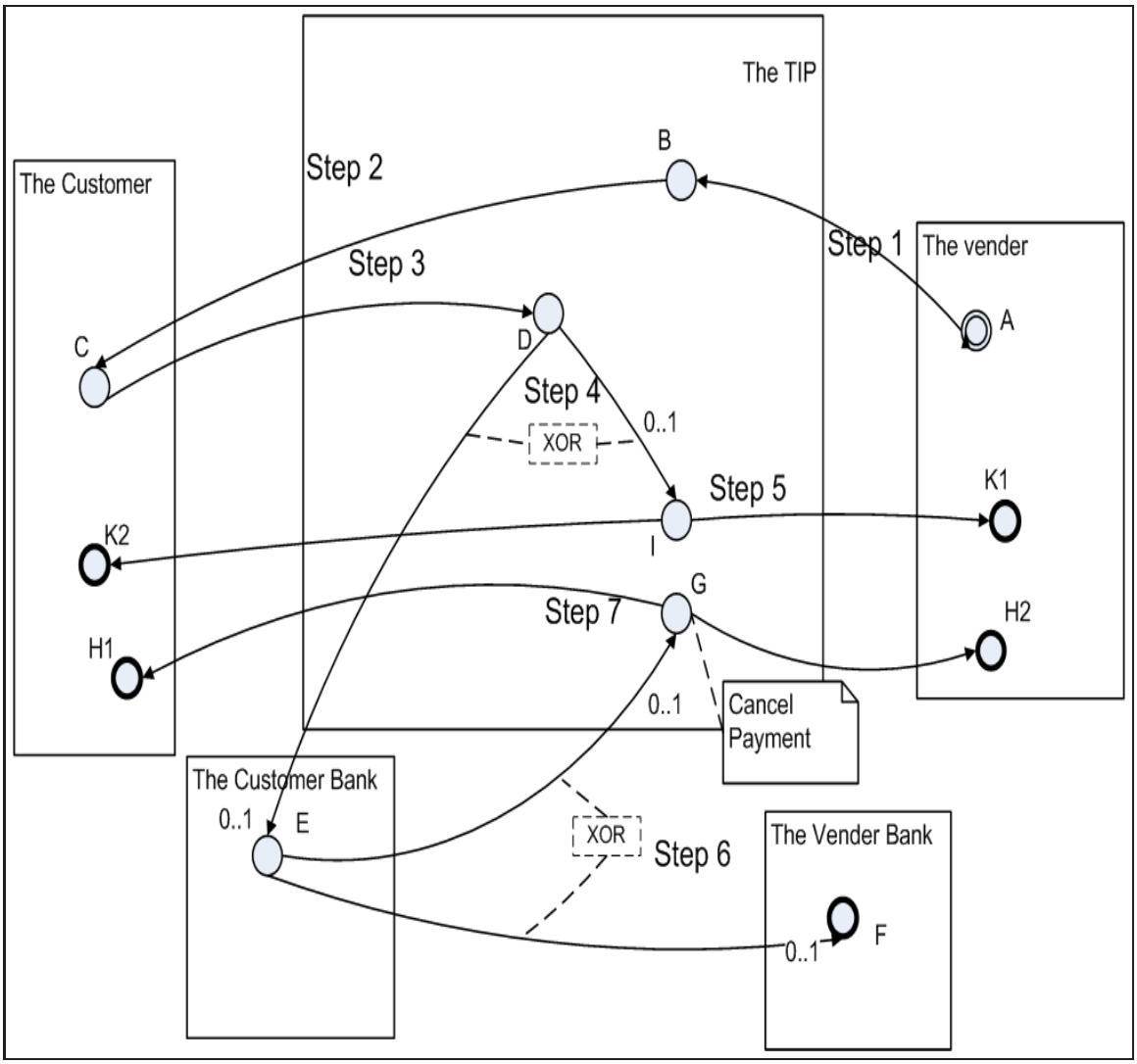

Figure 4.2: The Payment Transfer architecture in DTIM.

4.3 Formalizing the Modeling: the Frame Contract

Our fundamental constraint is based on the simple economic model just mentioned. Let us assume a pair of business parties are going to trade within this system applying the concept of fair exchange. The vendor has a set of goods that the customer wants to buy; the customer has the amount of money and asks for the goods. From a point of view of commerce, they are both expecting the value of their properties to grow after the trading. Therefore they both agree to keep the value of the goods as equal to the amount of the money for exchange during the transaction of transfer as a part of their contract. This agreement is a **frame contract** and is more general than the particular contract. By formalizing it, we present this frame contract as having the following constraints.
We set

- Time $t \in \{Begin, A, B, C, D, E, F, G, H1, H2, I, K1, K2, End\}$ as the point in time after the named certain transaction;

  Within set Time, we describe two time points: Begin is before transaction A, and End is after F, H1, H2, K1 or K2. They are the terminal points of the process.

- Item i: Goods to be transferred;

- Actor $a \in \{Customer, Vendor, TIP\}$;

- Cash $c(a, t)$: Integer (in dollars) which is the cash that actor a has at time t;

- Property $p(a, t)$: The total value of the properties that actor a has at time t;

- Value $v(i)$: Maps an item onto an integer value (in dollars);

- Define Net Fortune $n(a, t)$: $n(a, t) = c(a, t) + p(a, t)$.

From a commercial point of view, the customer and the vendor are probably both expecting to benefit from the trade. That is, in the end they expect the value of their fortune to grow through the trade, or at least to stay stable. Thus we obtain

Constraint 1(Actors do not lose net fortune at the terminal position.)

$$\forall a, n(a, End) \geq n(a, Begin)$$

We are more interested in fraud than in changes in net fortune due to different valuations of goods. We have modeled the value of the goods $v(i)$ as not time dependent. Indeed the value of the goods can change after the trading and the transfer of the money, but the value changes can be considered as another commercial process which is independent from this trade and insignificant to our purposes. Also we do not consider fees of TIP.

Another important assumption of our model is that it is a closed model, in which we have a conservation of cash and goods. That is, the cycling of the transfer is closed and the sum value of the cash and goods is a conservation of value. We do not consider the consistency of the transfer, which means no cash or goods can be lost during the transfer – all transactions should promise the items either in the hand of the source party or in the hand of the target party. We model this as:

Constraint 2 (Conservation of cash and goods)

$$\sum_a c(a, Begin) = \sum_a c(a, End)$$

$$\sum_a p(a, Begin) = \sum_a p(a, End)$$

From this follows a conservation of the sum of Net Fortune:

$$\sum_a n(a, Begin) = \sum_a n(a, End)$$

Both constraints can only be fulfilled if the fortune of all actors after the transaction is the same as before the transaction. Therefore we have the proposition

Proposition 1

From Constraint 1 and Constraint 2, it follows that

$\forall a, n(a, End) = n(a, Begin)$.

Proof

If no net fortune of a party has a decreasing value, then the conservation of the sum of net fortune demands that no net value of a party has an increasing value.

Some explanation is needed here. From 4.2.1 we have $F, H1, H2, K1$ and $K2$ before the time point *End*. Among them only transaction F describes the successful transfer. By all the other transactions, $n(Customer, End) = n(Customer, A) > n(Customer, Begin)$ and $n(Vendor, End) = n(Vendor, A) < n(Vendor, Begin)$ because the vendor delivers his items after the time point *Begin*. To attach a constraint to **Proposition 1**, we assume in the case of a failing transfer, there will be another transfer after time point *End*. That can be another payment transfer start from transaction A again, or a return shipment in which the customer returns the items to the vendor. In this case we regard $n(a, End) = n(a, A)$ as a specific situation for **Proposition 1**.

Since this proposition underlies assumptions of all the parties involved, this agreement is seen as a **frame contract** under the conditions of a conservation model and is less specific than the particular contract but contains a vital interest of the parties involved. If this contract is violated, then there is a risk of fraud.

As we set the value of the vendor's items as $v(i)$, we have the other group of equations under a successful fair exchange:

$n(Customer, Begin) = c(Customer, Begin) + p(Customer, Begin)$,

$n(Customer, End) = c(Customer, End) + p(Customer, Begin) + v(i)$,

and similarly

$n(Vendor, Begin) = c(Vendor, Begin) + p(Vendor, Begin)$,

$n(Vendor, End) = c(Vendor, End) + p(Vendor, Begin) - v(i)$.

Therefore because of Proposition 1, we have

$c(Customer, Begin) - c(Customer, End) = v(i)$,

$c(Vendor, End) - c(Vendor, Begin) = v(i)$.

Under these equations are assumptions that (1) both business parties have to agree that the cash exchanged in the transfer process reflects the value of the items and (2) they are equal to each other in a fair exchange.

Therefore we have the following corollary:

Corollary 1 of **Proposition 1**

In a fair exchange, the business parties agree that the cash transferred equals the value of the items.

$c(Customer, Begin) - c(Customer, End) = v(i) = c(Vendor, End) - c(Vendor, Begin)$

On the other hand, because the parties are involved in the transfer process, their net fortune must be changed in some steps of the process. Therefore $n(a, t)$ fluctuates during the process and returns to the value at the beginning. According to the profit risk mentioned in Section 2.4, we have the following proposition:

Proposition 2

$\forall t$, risks exist if $n(a, t) < n(a, Begin)$.

It is obvious that **Proposition 2** fulfils **Constraint 1**. Also we can tell in our economic model, because vendors deliver goods first, they are always under risk before they finally get the funds.

In addition, according to the frame contract agreement, we make the following declaration:

Declaration 1

In a close model, a fair exchange exists as long as $n(a, Begin) = n(a, End)$.

Also we have another corollary:

Corollary 2 of **Proposition 2**

If $n(a, Begin) <> n(a, End)$, a fraud exists in the transfer process that breaks the fair exchange.

To ensure the fair exchange, we need to detect any possibility of $n(a, Begin) <> n(a, End)$.

4.4 Analysis of the Modeling

Our analysis strategy is to trace all the possible roads of the transfer process. If we find in any road terminal, a result $n(a, Begin) <> n(a, End)$ exists, then we can declare a

risk of fraud.

4.4.1 Simplifying the model

To trace the possible roads in the transfer process, we test our constraints from the last section against the DTIM diagram. For the sake of simplicity we base the analysis on a couple of natural assumptions.

$p(Customer, Begin) = 0$,

$c(TIP, Begin) = 0$, $p(TIP, Begin) = 0$,

$c(Vendor, Begin) = 0$ and $p(Vendor, Begin) = v(i)$.

We observe

$n(Customer, Begin)$=$c(Customer, Begin) = v(i)$,

$n(TIP, Begin) = 0$, $n(Vendor, Begin) = v(i)$;

We would like to find

$n(Customer, End) = p(Customer, End) = v(i)$,

$n(TIP, End) = 0$,

$n(Vendor, End)$=$c(Vendor, End) = v(i)$.

According to 4.3, a fraud exists in a succeeding road if $n(TIP, End) <> 0$ or $n(Customer, End) <> v(i)$ or $n(Vendor, End) <> v(i)$; or it exists in a failing road if $n(TIP, End) <> 0$ or $n(Customer, End) <> 2v(i)$ or $n(Vendor, End) <> 0$.

4.4.2 Status of the Accounts

In every transaction, there are different operation behaviors. To design the system and implement the business logic within each transaction, we express parties' behaviors in a formula. Also, in the formula the input from former transactions is seen as a variable. To illustrate, assuming in an imaginary transaction X, the customer has two behaviors : 1) obtain the incoming data from a former transaction; and 2) add an asset to the source data. We use a variable transaction $former$ to describe the transaction one step before X. In a formula we can describe it as $X : n(Customer, X) = asset + n(Customer, former)$. Once

the business logic is decided, the stable behaviors in each transaction can be expressed in a formula in which the source data and output are variables. Therefore we can list all the formula for all the parties at all the time points of the process. Also, there is a formal relationship between the frame contract and the UMM: at a point in time $Begin$, all the parties have an initial state. We can calculate their account states at each time point according to the transactions formula. The states at terminal time point End are paramount for the analysis. In conclusion, the sequence of the transactions decide the terminal states of the parties. We list all the formula according to the UMM constraints.

(1) $Begin$: $n(Customer, Begin) = v(i)$, $n(TIP, Begin) = 0$,
$n(Vendor, Begin) = v(i)$;

(2) A : $n(Customer, A) = n(Customer, former) + v(i)$, $n(TIP, A) = n(TIP, former)$,
$n(Vendor, A) = n(Vendor, former) - v(i)$;

(3) B : $n(Customer, B) = n(Customer, former)$, $n(TIP, B) = n(TIP, former)$,
$n(Vendor, B) = n(Vendor, former)$;

(4) C : $n(Customer, C) = n(Customer, former)$, $n(TIP, C) = n(TIP, former)$,
$n(Vendor, C) = n(Vendor, former)$;

(5) D : $n(Customer, D) = n(Customer, former)$, $n(TIP, D) = n(TIP, former)$,
$n(Vendor, D) = n(Vendor, former)$;

(6) E : $n(Customer, E) = n(Customer, former)$, $n(TIP, E) = n(TIP, former)$,
$n(Vendor, E) = n(Vendor, former)$;

(7) F : $n(Customer, F) = n(Customer, former) - v(i)$, $n(TIP, F) = n(TIP, former)$,
$n(Vendor, F) = n(Vendor, former) + v(i)$;

(8) G : $n(Customer, G) = n(Customer, former)$, $n(TIP, G) = n(TIP, former)$,
$n(Vendor, G) = n(Vendor, former)$;

(9) $H1$: $n(Customer, H1) = n(Customer, former)$, $n(TIP, H1) = n(TIP, former)$,
$n(Vendor, H1) = n(Vendor, former)$;

(10) $H2 : n(Customer, H2) = n(Customer, former)$, $n(TIP, H2) = n(TIP, former)$, $n(Vendor, H2) = n(Vendor, former)$;

(11) $I : n(Customer, I) = n(Customer, former)$, $n(TIP, I) = n(TIP, former)$, $n(Vendor, I) = n(Vendor, former)$;

(12) $K1 : n(Customer, K1) = n(Customer, former)$, $n(TIP, K1) = n(TIP, former)$, $n(Vendor, K1) = n(Vendor, former)$;

(13) $K2 : n(Customer, K2) = n(Customer, former)$, $n(TIP, K2) = n(TIP, former)$, $n(Vendor, K2) = n(Vendor, former)$;

From the above formula we can see that these transactions in the business process mainly exchange the information, but only transactions E, F, and G were involved in the funding transfer. As we will not consider the security of the information exchange process, we focus on these funding transfer transactions.

4.4.3 Tracing the fraud by DTIM

From the features of the DTIM diagram, it is obvious that a "sequence" exists between the transaction steps. A sequence means a set of transactions, based on time and related to each other. Steps are the message exchanges between transactions. A sequence is also based on the order of the steps. A previous step in the sequence will finish before its following step (it is represented by the arrow line in the DTIM diagram) starts; the source of the previous step goes ahead of the target of the following step. If we trace the steps, we find that with this logic this transfer process consists of different "roads" of sequences.

To explain this, we setup a series of sequences for the transfer process. For example, in Figure 4.2 as $Step2 : B- > C$ is the previous step of $Step3 : C- > D$. Then the source transaction of Step 2 (which is B) is ahead of the target transaction of Step 3 (which is D). In a sequence Z, we describe this as $seqZ = B- > C- > D$. In addition, as the different conditions lead to a different sequence, we can build a set of sequences to cover all the situations in the application.

Therefore the original application can be expressed as $SetA\{Seq1, Seq2, Seq3\}$, in which

$Seq1 = Begin- > A - B- > C- > D- > I- > K1(K2)- > End$

$Seq2 = Begin- > A - B- > C- > D- > E- > F- > End$

$Seq3 = Begin- > A - B- > C- > D- > E - G- > H1(H2)- > End$

In each transaction, the transaction behaviors of funding are fixed, as described in 4.4.2. Also because we have the initial states of the parties, we can calculate the result of each sequence, and we can check whether the results of the sequence obey fairness rule 4.3.

To illustrate, assume P is the set of fair exchange result

P= $\{(n(Customer, End) = v(i),\ n(TIP, End) = 0, n(Vendor, End) = v(i)),$

$(n(Customer, End) = 2v(i),\ n(TIP, End) = 0, n(Vendor, End) = 0)\}$

We use $Result(i)$ to express the result of sequence i. Because the start states in our case are: $Begin$: $n(Customer, Begin) = v(i)$, $n(TIP, Begin) = 0$, $n(Vendor, Begin) = v(i)$, after calculating the sequence we have $Result(i) \in P, i \in SetA$.

4.5 Risk Analysis

In considering the profit risk, one of the security concerns is recourse. A recourse provides a chance for observing the transfer direction. By the end of the process, if the customer requires a recourse on bankC, he can receive a payment back. As any transaction working on the transfer is changing the states of the parties, the transaction for a recourse should be included in the risk analysis. Therefore the business logic extends to a new step:

- **Step 8** the bankC sends a recourse notice to the bankV [STOP].

Adding the recourse transaction into the DTIM diagram, we have a new payment transfer architecture, as shown in Figure 4.3.

Also we extend the UMM constraints and the formalization as follows:

```
Transaction > L
```

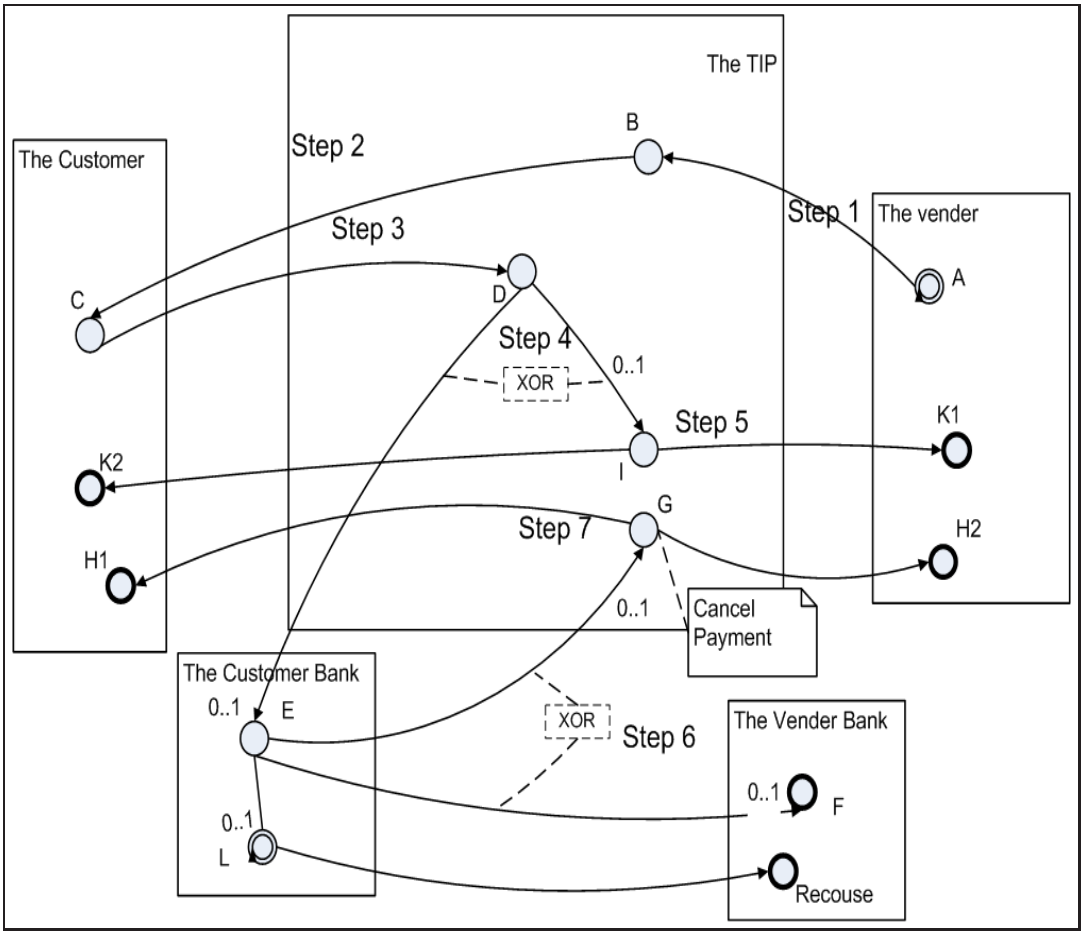

Figure 4.3: The Payment Transfer Architecture in DTIM–extended.

```
    total:Integer 1..1
    account1: account 1..1
Transaction > Recourse
    total:Integer 1..1
    account2: account 1..1
```

Step 8 L to $Recourse$: Output: $target.total = source.total$

SideEffect: $account1.net = account1.net + total$

SideEffect: $account2.net = account2.net - total$

- Time $t \in \{Begin, A, B, C, D, E, F, G, H1, H2, I, K1, K2, End\}$ as the point in time after the named certain transaction;

We add a new sequence into $SetA$ as $SetA' = \{Seq1, Seq2, Seq3, Seq4\}$, in which we have:

$Seq4 = Begin- > A- > B- > C- > D- > E- > F - L- > Recourse- > End$

Obviously, from $Seq4$ we can calculate the $n(Vendor, End) = 0$ and $n(Vendor, Begin) > n(Vendor, End)$, which is still risky for the vendor. Although $Result(Seq4) \in P$, from the DTIM diagram, the recourse transaction happens when the customer requires it, but the transaction does not work with TIP or the vendor. This makes the recourse transaction out of the control of TIP, and neither the payment system nor the vendor can notice it on time. The worst thing is, for the vendor, the account record information from TIP may differ from bankV's records. From the DTIM diagram we can see that no message exchange keeps them synchronized.

From this analysis, we can say that the recourse transaction may give an unlimited right to the customer, by which he can receive a refund without the TIP or the vendor taking notice. Although this recourse transaction may not break the fairness of the transfer, it can still make bankV's account information conflict with the information provided by TIP.

4.6 Summary

In this chapter, we modeled the current system and analyzed the modeling. We set up the constraints and the context of the model as the applicable conditions so that the model can be critically defined and formalized. As shown in the resulting model, all the end states are in the set of "fair" results. Therefore, we prove that the current system works as a fair exchange system. However, the recourse transaction is not included in the mechanism of TIP. By applying the recourse transaction, the customer can keep the items from the vendor longer than he should and thus puts the vendor in a profit risk. Also, from the analysis as shown in the DTIM diagram, the recourse transaction does not relate to TIP, which means no message is sent to TIP. It can cause a conflict between the information from bankV's account and the information provided by TIP. In the next

chapter, we address these problems with a new model that updates the funding transfer function, and we analyze the updated model with our form-oriented analysis method.

5

A New Design and its Security Analysis

In the last chapter we have formally modeled the current payment system and presented our security issues of this system. In this chapter we discuss applying form-oriented analysis on a future design as a tool for predicting and describing the security issue from business logic.

5.1 Reasons for Updating the Model

As we mentioned in the last chapter, the current AARN payment system works mostly for information exchange. The security issue for the current system is rarely at the business level and often on the implementation level. Moreover, recourse transaction is excluded in the current payment transfer, which not only confuses the account record of the vendor but also blocks the response of TIP for notifying the failing transfer to both of the business parties. Besides these, a more important problem of the effectiveness of the system emerges when the number of business parties increases. Strictly speaking, it is not a security issue but a problem of implementation. The system has the authority for using the parties' bank accounts to manage the payment process. As the parties have their own bank accounts respectively, it is very hard for the AARN system to connect with a variety

of different banks in order to get authorization to transfer funding from the account of a business partner. If we translate this condition in our model, according to the form oriented analysis feature, the model can be seen as one example for multi-instances. The multi-business parties can have multiple banks, but TIP can not support all the banks from a practical point of view.

Therefore, we can re-build the architecture with a banking subsystem. We include the recourse transaction into this subsystem because it is also working on funding exchange. In the banking subsystem, a bank account of TIP (called bankTIP) is in charge of contacting the various banks. According to the order of the TIP, the bankTIP first takes a direct debit payment from a bankC, then transfers the funding to the bankV through a direct credit payment.

5.2 The Payment Transfer Architecture

In this section we describe the updated design of the payment system. In the new design, the scenario of the business is still the same as in Chapter 4. We use the same constraints defined in Chapter 4 as well. But we define a bankTIP as the bank account of TIP to help the TIP manage the funding. BankTIP is in an intermediate bank which is in charge of all the connections to the various banks for the different business parties. This star topology architecture utilizes the existing banking network to transfer payment in the new system.

During the design process, the intermediate bank requires a new feature for the payment system for marketing purposes. The payment must be live to the business parties. That is, the transfer process will have no delays after the funding comes from bankC. Then funding should be transferred right after the direct debit payment so that the whole system works more efficiently and provides real-time payment records to the business parties.

5.2.1 The Updated Simple Economic Model

In the updated model, a payment transfer process involves a vendor, a customer, a TIP, and their banks respectively. Like the scenario in Chapter 4, TIP first receives a invoice from the vendor and then he asks the customer to approve the invoice. The customer accepts the invoice and replies to TIP with confirmation information. TIP then contacts the banking subsystem in charge of moving the funding between banks. The banking subsystem receives the order of payment from the TIP and executes the order; if there is an error message, the banking subsystem forwards it to the TIP. Here is a description of the transfer steps:

- **Step 1** The vendor sends an invoice to the TIP, [GOTO **Step 2**];
- **Step 2** The TIP forwards the invoice to the customer and records it as a log record, [GOTO **Step 3**];
- **Step 3** The customer accepts the invoice and sends out a payment instruction to the TIP, [GOTO **Step 4**];
- **Step 4** The TIP records the payment instruction and checks it with the invoice record. If they match, TIP will send the payment instruction to the intermediate bank, [GOTO **Step 6**]; otherwise it will mark the payment as failed, [GOTO **Step 5**];
- **Step 5** The TIP rejects this failed payment instruction and notifies both parties [STOP].
- **Step 6** The intermediate bank executes the banking processing subsystem (see Figure 5.3) to transfer the funds from the customer's bank to the vendor's bank through the TIP account. If it is successfully conducted, the process is finished [STOP]; otherwise, the intermediate bank will send a failure notice to the AARN payment system, [GOTO **Step 7**];
- **Step 7** The TIP sends a failure notice to both of the parties [STOP].

5.2.2 The Payment Transfer architecture

As in Chapter 4, first we setup UMM structures according to the economic model, as this:

```
Transaction > A
    total: Integer 1..1
    invoiceID: invoice 1..1
Transaction > B
    total: Integer 1..1
    invoiceID: invoice 1..1
Transaction > C
    total: Integer 1..1
    paymentID: payment 1..1
Transaction > D
    total: Integer 1..1
    invoiceID: invoice 1..1
    paymentID: payment 1..1
    bankAccountID1: 1..1
    bankAccountID2: 1..1
Transaction > E
    total: Integer 1..1
    bankAccountID1: 1..1
    bankAccountID2: 1..1
Transaction > F
Transaction > G
    total: Integer 1..1
    invoiceID: invoice 1..1
    paymentID: payment 1..1
Transaction > H1
    total: Integer 1..1
    paymentID
Transaction > H2
    total: Integer 1..1
    invoiceID: invoice 1..1
Transaction > I
    total: Integer 1..1
    invoiceID: invoice 1..1
    paymentID: payment 1..1
Transaction > K1
    total: Integer 1..1
    paymentID: payment 1..1
Transaction > K2
    total: Integer 1..1
    invoiceID: invoice 1..1
```

According to the business logic described in 5.2.1, we setup the UMM constraints for the new architecture as:

Step 1 A to B: Output: $target.total = source.total$

//the amount of money from the invoice recorded into the system

$target.invoiceID = source.invoiceID$

Step 2 B to C: Output: $target.total = source.total$

//the amount of the invoice equals the money the buyer will pay

Step 3 C to D: Output: $target.total = source.total$

$target.paymentID = source.paymentID$

Step 4 D to E: Flow: $source.total = B \rightarrow allinstances() \rightarrow select(invoiceID).total$

Output: $target.total = source.total$

$target.bankAccountID1 = source.bankAccountID1$

$target.bankAccountID2 = source.bankAccountID2$

D to I: Flow: else Output: Fail message

Step 5 I to K1: Output: $target.invoiceID = source.invoiceID$

message(fail)

I to K2: Output: $target.paymentID = source.paymentID$

message(fail)

Step 6 E to F: Success message

E to G: Output: $target.total = source.total$

Step 7 G to H1: Output: $target.invoiceID = source.invoiceID$

message(fail)

G to H2: Output: $target.paymentID = source.paymentID$

message(fail)

The DTIM diagram is shown in Figure 5.1.

From the above models and diagram we can see that most steps of the new architecture are identical to the current system. As we have discussed the current architecture, we know that these steps work for information exchange, and the states of these transactions are reliable. On the other hand, the banking subsystem is working on transferring funds as the most important part in the security analysis business logic.

5.2.3 The Banking Subsystem

As the banking processing subsystem is a key point in the architecture, from now on we consider this subsystem with especial focus on the working steps of the funding transfer

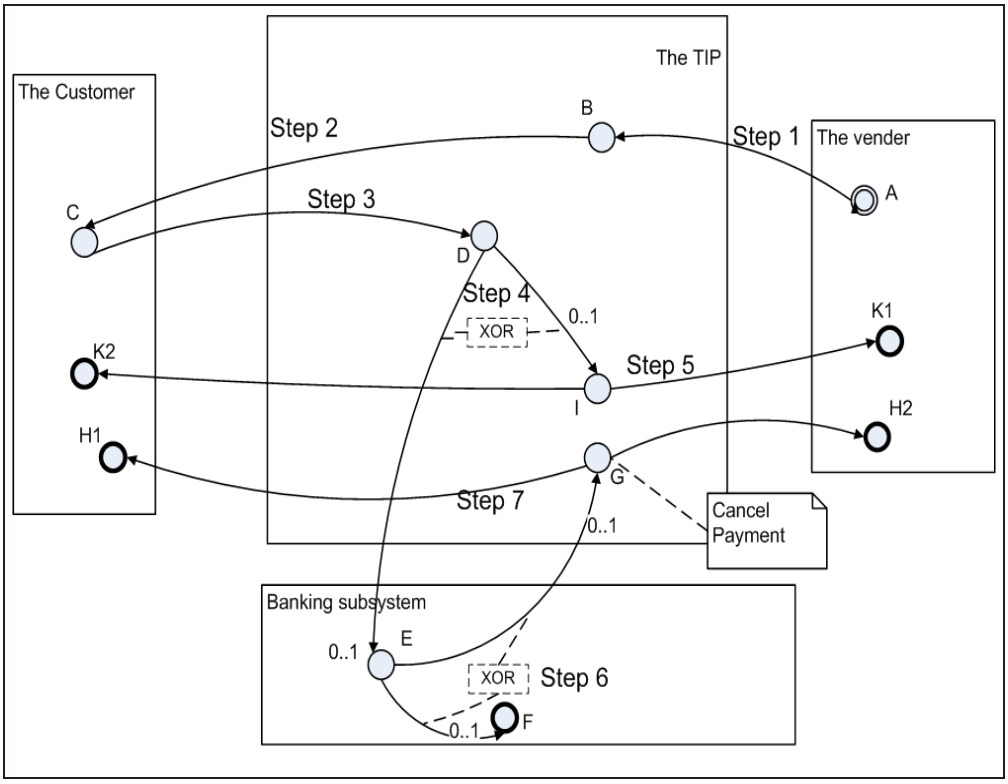

Figure 5.1: The Payment Transfer architecture in DTIM.

process within the subsystem. Also, as mentioned in 5.1, we include the recourse process into the subsystem. The business logic for the subsystem is:

Step E1 The intermediate bank sends a request of D/D instruction to the customer's bank, [GOTO **Step E2**];

Step E2 The customer's bank accepts it and responds with a D/D payment instruction [GOTO **Step E3**] with the proviso that within the next two days it may ask for a recourse [GOTO **Step E5**]; or the customer's bank rejects this request and the intermediate bank records the exceptional condition [Stop];

Step E3 The intermediate bank sends a request of D/C instruction to the vendor's bank [GOTO **Step E4**];

Step E4 The vendor's bank confirms the request [Stop]; otherwise it rejects the requirement and the intermediate bank records the exceptional condition [Stop].

Step E5 The customer's bank asks for a recourse and the intermediate bank accepts the requirement [Stop].

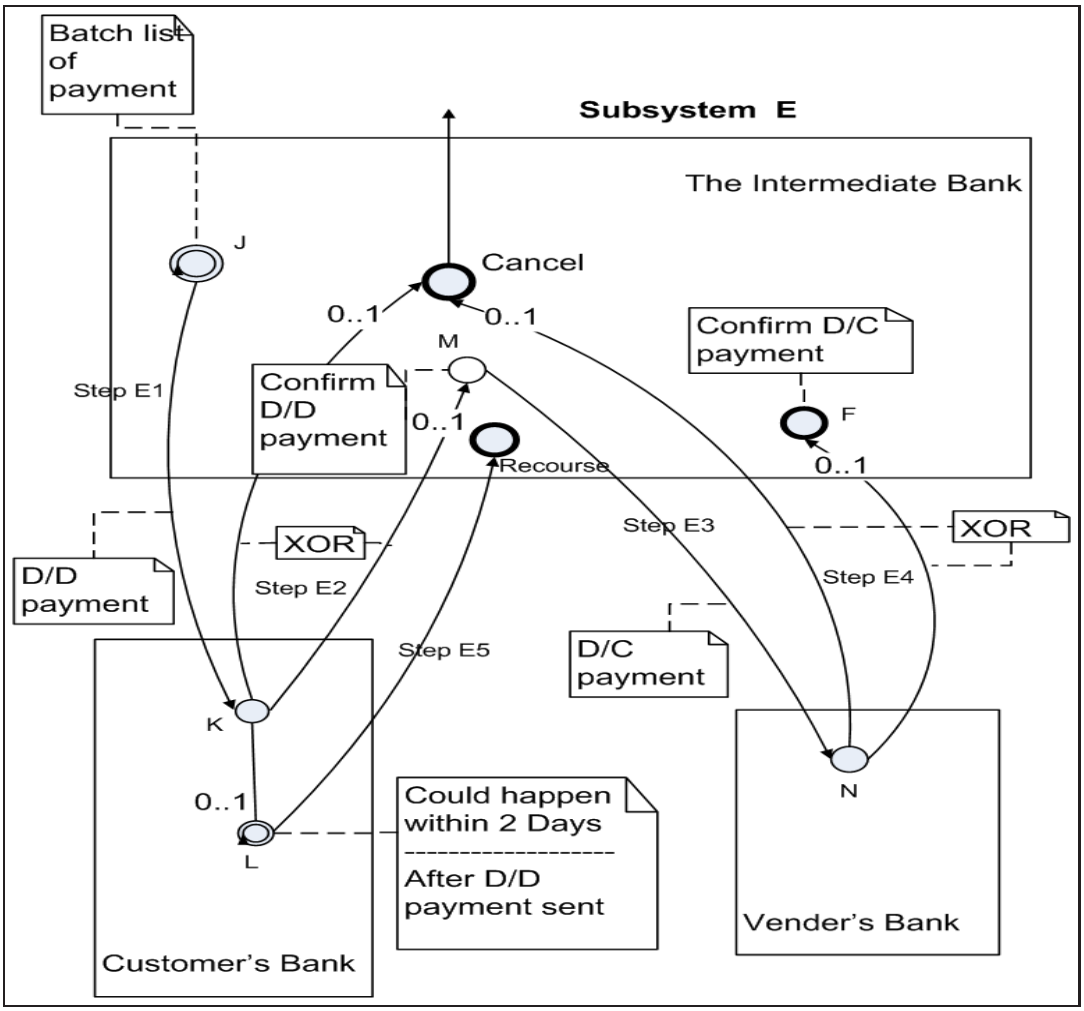

Figure 5.2: The banking processing subsystem.

The recourse option of **Step E2** requires further discussion. For all D/D payments, the banking industry allows the payer (the customer in this instance) to retain a right of recourse for up to two days. Therefore, if any problem arises with the later stages of the financial transaction, the customer's bank can reverse the D/D instruction – if it acts rapidly enough.

Only a D/D payment has a right of recourse. A fund transfer by D/C payment to the vendor's account is irrevocable, so it can be withdrawn by the owner of the account as soon as their bank has accepted the D/C request.

The UMM data structure for this business logic is:

```
Transaction > J
    total:Integer 1..1
    account1: account 1..1
    account2: account 1..1

Transaction > K
    total:Integer 1..1
    account1: account 1..1
Transaction > M
    total:Integer 1..1
Transaction > N
    total:Integer 1..1
    account2: account 1..1

Information > Account
    net:Integer 1..1
Transaction > F


Transaction > Cancel
    total:Integer 1..1
Transaction > L
    total:Integer 1..1
    account1: account 1..1
Transaction > Recourse
    total:Integer 1..1
```

According to the business logic, we setup the UMM constraints for the subsystem as:

Step E1 J to K: Output: $target.total = source.total$

Step E2 K to M: Output: $target.total = source.total$

SideEffect: $account1.net = account1.net - total$

K to $Cancel$: Output: message(failed)

Step E3 M to N: Output: $target.total = source.total$

Step E4 N to F: Output: $target.total = source.total$

SideEffect: $account2.net = account2.net + total$

N to $Cancel$: Output: message(fail)

Step E5 L to $Recourse$: Output: $target.total = source.total$

SideEffect: $account1.net = account1.net + total$

In an early stage in the design of the AARN payment system, the intermediate bank suggested an option of providing a more attractive feature to the business parties. The

bank would transfer the funds to the vendor's account as soon as it receives the D/D payment from the customer, without any delay in the TIP account. The benefit is that the transfer process will become more efficient and therefore more attractive to the vendors. As we have mentioned, the vendors are taking the bigger risk in the credit transfer, so obviously the vendors would welcome this feature. However, we can prove that this feature creates a financial risk to the TIP.

5.3 Analysis of the Modeling

Analysis of the modeling requires formalizing it as in Chapter 4. However, as the constraints of the scenario change little and the formalization of the model follows the same steps as Chapter 4, we do not repeat it here. All the constraints and propositions are presented in Chapter 4. Now we start the analysis for the subsystem.

5.3.1 Status of the Accounts

Listed the below is the formula for the banking subsystem.

(1) $Begin$: $n(Customer, Begin) = v(i)$, $n(TIP, Begin) = 0$,
$n(Vendor, Begin) = v(i)$;

(2) J : $n(Customer, J) = n(Customer, former) + v(i)$,
$n(TIP, J) = n(TIP, former)$, $n(Vendor, J) = n(Vendor, former) - v(i)$;

(3) K : $n(Customer, K) = n(Customer, former)$,
$n(TIP, K) = n(TIP, former)$, $n(Vendor, K) = n(Vendor, former)$;

(4) M : $n(Customer, M) = n(Customer, former) - v(i)$,
$n(TIP, M) = n(TIP, former) + v(i)$,
$n(Vendor, M) = n(Vendor, former)$;

(5) N : $n(Customer, N) = n(Customer, former)$,
$n(TIP, N) = n(TIP, former) - v(i)$,

$n(Vendor, N) = n(Vendor, former) + v(i)$;

(6) $F : n(Customer, F) = n(Customer, former)$,
$n(TIP, F) = n(TIP, former)$, $n(Vendor, F) = n(Vendor, former)$;

(7) $Cancel : n(Customer, Cancel) = n(Customer, former)$,
$n(TIP, Cancel) = n(TIP, former)$, $n(Vendor, Cancel) = n(Vendor, former)$;

(8) $L : n(Customer, L) = n(Customer, former)$,
$n(TIP, L) = n(TIP, former)$, $n(Vendor, L) = n(Vendor, former)$;

(9) $Recourse : n(Customer, Recourse) = n(Customer, former) + v(i)$,
$n(TIP, Recourse) = n(TIP, former) - v(i)$,
$n(Vendor, Recourse) = n(Vendor, former)$;

5.3.2 Tracing the Fraud by DTIM

As the requirements for the application changed, the sequence of the steps changes. In addition, we may also create new steps and transactions or remove existing ones. In our case, the new requirement asks to change the sequence of StepE3 running right after StepE2, so we can build a new set of sequences, SetB, for the new transaction steps.

$SetB\{Seq5, Seq6, Seq7, Seq8\}$

$Seq5 = Begin- > J- > K- > M- > N- > F- > End$

$Seq6 = Begin- > J- > K- > Cancel- > End$

$Seq7 = Begin- > J- > K- > M- > N- > Cancel- > End$

$Seq8 = Begin- > J- > K- > M- > N - L- > Recourse- > End$

$Seq8$ implement the requirement of running StepE3 exactly after StepE2.

Assume P is the set of fair exchange result,

P= $\{(n(Customer, End) = v(i), n(TIP, End) = 0, n(Vendor, End) = v(i))$,
$(n(Customer, End) = 2v(i), n(TIP, End) = 0, n(Vendor, End) = 0)\}$

We use $Result(i)$ to express the result of sequence i, for example $Result(Seq5) = n(a, F)$, $Result(Seq7) = n(a, Cancel)$, $a \in \{Customer, Vendor, TIP\}$. Because the

start states in our case are: $Begin$: $n(Customer, Begin) = v(i)$, $n(TIP, Begin) = 0$, $n(Vendor, Begin) = v(i)$, after calculating the sequence we have $Result(Seq7) not \in P$, $Result(Seq8) not \in P$. According to Proposition 2 and its Corollary 2, there are risks in these sequences. We call the sequences ending in a fair exchange result as correct roads, those not ending in a fair exchange result as risky roads. In our case we call the risks in $Seq7$ and $Seq8$ as **Risk 1** and **Risk 2** respectively. We will discuss the solution to these risks in the next section.

5.4 Solution of the Risks

To locate the risks, we set the states of the transactions in a correct road as correct states. If a correct road and a risky road start from the same point, we find the furthest transaction on a risky road, which has the same states as the correct states. Then the risk exists as a result of the business logic problem between this transaction and the next transaction, otherwise the correct states can not ensure the road is correct. For example, $Seq5$ and $Seq7$ share the same route until transaction N, then the Risk 1 should result in N and $Cancel$. For Risk 2, as $Seq5$ and $Seq8$ share the same route until N, we can say Risk 2 is because of the sequencing between N and L.

5.4.1 Solution for Risk 1

In the design process, it is harder to locate the risks than find out what the risk is. As for Risk 1, the behaviors of transaction $Cancel$ do not fit the business logic after transaction N. The current transaction $Cancel$: $n(Customer, Cancel) = n(Customer, former)$, $n(TIP, Cancel) = n(TIP, former)$, $n(Vendor, Cancel) = n(Vendor, former)$ is for canceling the operations of transaction K. After transaction N, as the "former" behaviors changed, the behaviors of $Cancel$ should be changed as well. Therefore, in these two situations we need different $Cancel$ transactions. As a result, we separate the $Cancel$ as $Cancel - D/D$, $Cancel - D/C$. Within them, $Cancel - D/D$ is for canceling the D/D payment resulting from the behaviors of K; $Cancel - D/C$ is for canceling the

D/C payment resulting from the behaviors of N. The UMM model change for this is:

```
Transaction > Cancel-D/D
    total:Integer 1..1

Transaction > Cancel-D/C
    total:Integer 1..1
```

Also we change the constraints of the UMM but do not present them here to avoid repeating – they are similar to the previous constraints except the name of the transaction.

5.4.2 Solution for Risk 2

After analyzing Risk 2, we can see the problem is that L happens after N. The sequence of M following N is the requirement from the clients to the system, but the business logic shows that it conflicts with the mechanism of recourse.

Our solution for Risk 2 is a simple one. We ignore the new unavailable requirement and set the constraint that L must start before N, then $Seq8 = Begin- > J- > K- > M - L- > Recourse- > End$

According to this, we designed a timer system to handle the recourse process. A timer pauses the transfer for a period before a recourse expires. After the TIP's bank receives the funding from the customer, the transfer is paused temporally before the expiry time. We call this period the timing period. Within this period, if a recourse requirement from the customer's bank is sent to the TIP's bank, the timer records this requirement for the specific transfer. After the timing period, the timer checks the record of the transfer. If there is a recourse requirement, it notifies the TIP's bank to cancel the transfer because of the recourse request. Otherwise, after the timing, the timer notifies the TIP's bank to continue the transfer process.

The User Message Model for this logic is:

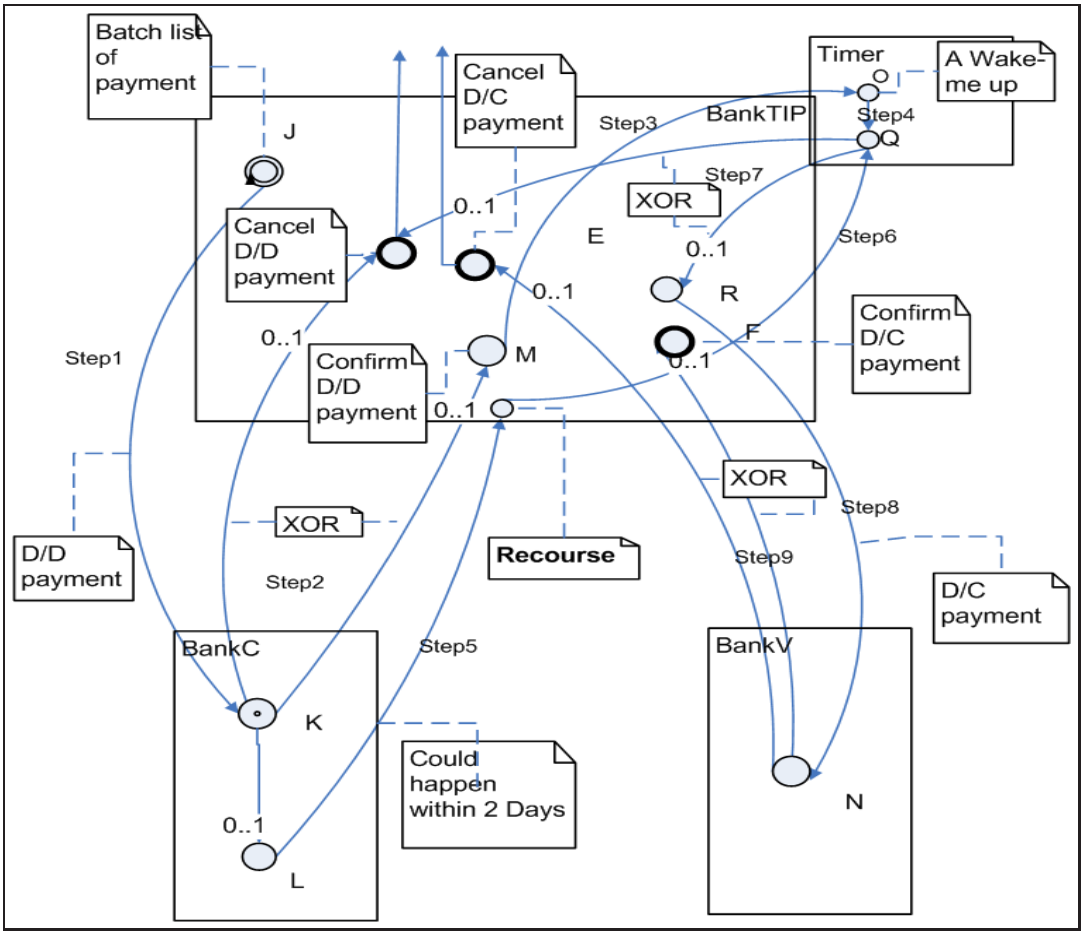

Figure 5.3: The banking processing subsystem–Recourse &Timer.

```
Transaction > J                 Information > Account
    total:Integer 1..1              net:Integer 1..1
    account1: account 1..1      Transaction > F
    account2: account 1..1
Transaction > K                 Transaction > O
    total:Integer 1..1              net:Integer 1..1
    account1: account 1..1      Transaction > Q
Transaction > M                     net:Integer 1..1
    total:Integer 1..1          Transaction > R
Transaction > N                     net:Integer 1..1
    total:Integer 1..1              account1: account 1..1
    account2: account 1..1
```

```
Transaction > Cancel-D/D        Transaction > L
    total:Integer 1..1              total:Integer 1..1
Transaction > Cancel-D/C            account1: account 1..1
    total:Integer 1..1          Transaction > Recourse
                                    total:Integer 1..1
```

According to the business logic, we setup the UMM constraints for the subsystem as:

Step E1' J to K: Output: $target.total = source.total$

Step E2' K to M: Output: $target.total = source.total$

SideEffect: $account1.net = account1.net - total$

K to $Cancel - D/D$: Output: message(failed)

Step E3' M to O: Output: $target.total = source.total$

Step E4' O to Q: Output: $target.total = source.total$

Step E5' L to $Recourse$: Output: $target.total = source.total$

SideEffect: $account1.net = account1.net + total$

Step E6' M to Q: Output: $target.total = source.total$

Step E7' Q to R: Output: $target.total = source.total$

Q to $Cancel - D/D$: Output: message(fail)

Step E8' R to N: Output: $target.total = source.total$

Step E9' N to F: Output: $target.total = source.total$

SideEffect: $account2.net = account2.net + total$

N to $Cancel - D/C$: Output: message(fail)

There are some other solutions which, by applying our method, we can also analyze. We will not discuss them in our case study but leave them for future works. These solutions may be:

(1) Changing the object of recourse and the mechanism of recourse, adding in the recourse for direct credit payment. In this case the vendor will take the risk of recourse.

(2) Building credit through the TIP, or asking the business parties to leave a deposit on the TIP. This will change the *Begin* states of all the parties and thus change the model.

(3) Changing the business logic constraints, the vendor stopping delivery of items first. This would change the transactions, constraints, and *Begin* states of the model, and thus change the model.

5.5 Evaluation of the Modeling

Our formal modeling is limited in constraints and conditions of the model change. Although the model changes, or new models are created, we can apply this method to predict the business risks.

Our formal modeling is the first time involving form-oriented analysis in business logic analysis and also the first used in security analysis. There was business security analysis from a commerce point of view, but they are mostly work flow diagrams. Our modeling can formalize the business and be extended to UML as well as work for the future system design.

However, automatic form-oriented analysis tools are needed for drawing figures and converting the DTIM object symbols into detailed UML objects. It also should provide the detection of business logic. Business logic can be setup for different analysis. For example, in our case, we setup fairness n(a,Begin)=n(a,End) under a model of conservation on net fortune, as a constraint for the security analysis. Other features of security (confidentiality,

identifiability) can have other propositions and constraints. As the symbols are formal, it is not necessary to define data sets or other structures. With the design tools we can check them while we are drawing the diagram, so that the designers do not need to know the process of proof or inferential reasoning. The tools should provide auto-detection for the diagram while designing, so that risky designs can not be generated.

5.6 Summary

This chapter concerns a modified model for the AARN payment system. In this modified model, we try to solve the risk created by the independent action of the customer. We separate the banking transfer function as a subsystem and focus on the influence of the recourse process to this subsystem. Applying the same steps of form-oriented analysis as in Chapter 4, we find two risks in the modified model. The first risk is that, under the current recourse process, the TIP's bank can not transfer funding directly from the customer to the vendor without putting itself in a risky situation. The second risk is that the *Cancel* transactions act in different conditions, have different behaviors, and therefore we should define different transactions to describe them. Otherwise the semantics can not stay consistent on the predicate while keeping fairness on the logic.

6 Conclusion

In this chapter, we summarize the risk analysis in the business logic and, as the target of our analysis, we conclude our form-oriented analysis method. Next, we briefly discuss the result of the analysis in the case study of AARN payment system. Lastly, we give out projections on future research for form-oriented analysis in security.

6.1 Risks in the Business Processes

In any application design, risks are an attendant problem for the security of the application. Compared to the risk analysis in the system architecture processes, the risk analysis in the business logic is a very rare concern. After lessons from case to case, software engineers realized that even in the requirements from senior fellows in the mature industries conflicts and potential frauds can still exist. The reason for this is that human actions sometime automatically avoid the negative possibilities in a business, for example, nobody would pay -10 dollars. The staff can personally setup business rules to handle the conflicts. Unfortunately, the ones providing the business logic sometimes forget to specify what kind of business rules are in it. Computer software has not yet advanced so that it works like humans who can learn business and understand the risks through experience.

The risks in a logic level can not be found in the architecture or the implementation process, as they are on a different abstract level. It is impossible to clearly integrate the concepts in different levels into a single description using the predicate. Therefore, the response of computer scientists is to analyze and understand the risks hidden in business logic.

As the former chapters have illustrated, we work on the security analysis for business logic. In our case study, we explored the recourse process in a fair exchange environment and detected the risks in the fairness. As a result of the proof, we announce the risks for the recourse process which can challenge the business. These risks can lead to changes in business patterns.

6.2 Formal Security Analysis

In this thesis, we discussed the security analysis in business logic. Analysis in business logic actually is based on the theory of software engineering to reduce the risk on the implementation process of an application system. The security analysis in business logic level aims to understand the business for both the designers and the clients for security purposes. This analysis is working on the early stage of the functional specification and prototype to avoid further harm to the implementation. An effective security analysis can avoid wasting capital, time, and concentration. Our security analysis method mentioned in Chapters 4 and 5 is form-oriented analysis in which we applied the DTIM/UMM modeling language. The basic steps of our form-oriented analysis method are:

(1) define the business logic applied in the analysis;

(2) build constraints, propositions and other rules based on this business logic as the frame contract;

(3) formalize the state of all the transactions;

(4) examine every constraint, proposition or specific rule of the frame contract into the transactions;

(5) find out any conflict between the states of the parties and the frame contract.

In our case study, we define the fairness feature as a frame contract to the system. After the mathematical inference is clarified, the final constraints can be used to keep the fairness feature of the business. We set the initial state of each party in the *Begin* transaction, and we can determine the behaviors of each transaction. Therefore as long as we can decide the order of the transactions, we can calculate all the states of all the parties in each transaction.

As shown in Chapters 4 and 5, the sequence of the transactions can finally describe the route of the business logic. Different roads in the route of logic demonstrate different choices for the execution. If we can check the validity of the roads by the results of execution, we can tell that every state of the parties on the road obeys the frame contract. For those roads which are unavailable because of the conflict with the frame contract, we can trace the sequence and finally locate the risk transaction. In the case study in Chapter 5, the method not only detects the risk on the recourse process, it also locates the duplex meaning for transaction *Cancel*, and then stops a risky design before a further step of implementation started. According to the rules of software engineering, it is especially beneficial to find out the risks in a design as early as possible.

All in all, we carried out the analysis to find the risk transaction and explore the business logic under the specific conditions. We applied the form-oriented diagrams (DTIM), data structures and constraints (UMM)in this analysis and developed the formal proof for the analysis.

Our method of form-oriented analysis on security is a promising and novel for the security analysis on business logic. It detects the risks on the business logic level by checking the constraints on the sequences of transactions. The method works on the security analysis because form-oriented analysis is based on the request/response style, and the semantics of the method are simple but clear to formally describe the business. Underlying the method, the DTIM diagrams are most important for illustrating the sequences of transactions. The UMM data constraints are for formally defining the transactions.

The frame contracts are the conditions for the security analysis. This formal modeling for the business logic can be used as a basic discussion for the application from business logic level.

6.3 Future Research

Our aims in research on the business logic level were to present the business logic in a simple but clear method which is easy to understand and works for the communication between the clients and application designers. In DTIM diagrams we use simple symbols so that the clients can understand the abstract issues of the business; the benefit of the simple symbol is it enables the clients to avoid confusing the meaning of various kind of symbols, so that they can focus on the business logic itself. However, we have not tried the method in a real meeting with the clients, so the effectiveness of the method is still hypothetical. Only after it is used in different applications can we survey and statistic the effectiveness of this method.

In our case study, we did the security analysis only on the business logic level. Future security analysis projects would need a full security analysis including the architecture and the implementation processes. Implementing the secure design for the EDI data structures and standard (such as AS2) and analyzing the detail for transactions would enable construction of a formal security model for the payment system.

The work we have done provides a new direction for applying for form-oriented analysis. Form-oriented analysis works on diagrams and data structure definition for security purposes. It is tough design work, especially when the business logic becomes complex. Currently there is no specific design tool to facilitate such work for designers. In our case study, we used Microsoft Visio to create the DTIM diagram, but we can not integrate the UMM constraints into the diagram. We have to describe them separately, which may lead to confusion and misunderstanding for readers. A proper tool would provide two benefits: first, designers could setup the frame contract before starting the design. Then during the process of designing for business logic, the tool could help detect conflicts to the frame

contract on the design. The other benefit is reusing the frame contract. The constraints and concepts could be used in other case for the same security concern. When changes happens to business conditions, the concept patterns facilitate the security analysis by avoiding the repetition of the steps in this analysis.

The overall importance of the research is the novel exploration for the form-oriented analysis method on security issues. Using form-oriented analysis in an e-commerce security context is novel. In our case study, we use form-oriented analysis to detect the profit risks in a fair exchange. With this tool, we were surprised to learn that under the conditions for fairness, an improper requirement on the recourse transaction can lead to risky business. In the future, by adding into the other business logic such as legal issues, this case study can be extended to a whole new possible research area for B2B e-commerce.

Index

Bibliography

[1] January 2001 b2b: Quest for profitability. *Wired-whats next now*, 2001. Available on `http://www.whatsnextnow.com/special_sections/9.01_b2b/jan.b2b.html`, last update: 03/02/2001.

[2] Han Zhang William Zhu Gerald Weber Clark Thomborson. B2b e-commerce security modeling: A case study. Submitted to CIS06, Computational Intelligence and Security, November 3-6, 2006, Guangzhou, China., 2006.

[3] N. Asokan. *Fairness in electronic commerce*. PhD thesis, Waterloo, Ont., Canada, Canada, 1998.

[4] Shoup V. Waidner M Asokan, N. Optimistic fair exchange of digital signatures. *Selected Areas in Communications, IEEE Journal*, 18(4):593–610, 2000.

[5] Basel Committee on Banking Supervision. *Basel II: International Convergence of Capital Measurement and Capital Standards: a Revised Framework*. Bank for International Settlements, 2005. Available on `http://www.bis.org/publ/bcbs118.htm`, last update: November, 2005.

[6] Ann Bednarz. The art of balancing e-commerce processes, February 2005. Available on `http://www.networkworld.com/news/2005/020705specialfocus.html`, last update:07/02/05.

[7] Dun & Bradstreet. Trade payment analysis–australian trade payments continue to worsen, 2006. Available on `http://www.dnb.com.au/pdfs/general/TPA_2006_AU.pdf`, last update: 30/03/06.

[8] Stephen D. Brookes, C. A. R. Hoare, and A. W. Roscoe. A theory of communicating sequential processes. *J. ACM*, 31(3):560–599, 1984.

[9] Jery Bryans. Reasoning about xacml policies using csp. In *SWS '05: Proceedings of the 2005 workshop on Secure web services*, pages 28–35, New York, NY, USA, 2005. ACM Press.

[10] Jim Conallen. Modeling web application architectures with uml. *Commun. ACM*, 42(10):63–70, 1999.

[11] Cybersource Corporation. 7th annual online fraud report–online payment fraud trends merchant practices & benchmarks, February 2006. Available on `http://www.cybersource.com/resources/collateral/Resource_Center/whitepa%pers_and_reports/CYBS_2006_Fraud_Report.pdf`.

[12] Bart De Decker. Introduction to computer security. In *State of the Art in Applied Cryptography*, pages 377–393, 1997.

[13] Thuong Doan, Steven A. Demurjian, T. C. Ting, and Andreas Ketterl. Mac and uml for secure software design. In *FMSE*, pages 75–85, 2004.

[14] Dirk Draheim and Gerald Weber. Modeling submit/response style systems with form charts and dialogue constraints. pages 267–278, 2003.

[15] Dirk Draheim and Gerald Weber. *Form-Oriented Analysis-A New Methodology to Model Form-Based Applications*. Springer Verlag, 2004.

[16] Matthew B. Dwyer, Lori A. Clarke, and Kari A. Nies. A compact petri net representation for concurrent programs. In *ICSE*, pages 147–157, 1995.

[17] Paolo Falcarin and Maurizio Morisio. Developing Secure Software and Systems. *IEC Network Security: Technology Advances, Strategies, and ChangeDrivers*, 2004. Available on `http://softeng.polito.it/falcarin/docs/sec-report04.pdf`.

[18] Manfred Hauswirth, Mehdi Jazayeri, and Markus Schneider II. A phase model for e-commerce business models and its application to security assessment. In *HICSS*, 2001.

[19] C. A. R. Hoare. Communicating sequential processes. *Commun. ACM*, 21(8):666–677, 1978.

[20] Jurjens J Houmb, S.H. Developing secure networked web-based systems using model-based risk assessment and umlsec. In *Software Engineering Conference, 2003*, pages 488– 497, Dec 2003.

[21] Werner B. Joerg. A subclass of Petri nets as design abstraction for parallel architectures. *SIGARCH Comput. Archit. News*, 18(4):67–77, 1990.

[22] James Joshi, Walid G. Aref, Arif Ghafoor, and Eugene H. Spafford. Security models for web-based applications. *Commun. ACM*, 44(2):38–44, 2001.

[23] Sherrie Xiao Komiak and Izak Benbasat. Understanding customer trust in agent-mediated electronic commerce, web-mediated electronic commerce, and traditional commerce. *Inf. Tech. and Management*, 5(1-2):181–207, 2004.

[24] Nir Kshetri and Nikhilesh Dholakia. Determinants of the global diffusion of b2b e-commerce. *Electronic Markets*, 12(2), 2002.

[25] Susan Kuchinskas. Fraud chewing e-commerce profits, November 2005. Available on `http://www.internetnews.com/ec-news/article.php/3563061`.

[26] Vincent S. Lai and Bo K. Wong. Business types, e-strategies, and performance. *Commun. ACM*, 48(5):80–85, 2005.

[27] Torsten Lodderstedt, David A. Basin, and Jürgen Doser. Secureuml: A uml-based modeling language for model-driven security. In *UML*, pages 426–441, 2002.

[28] Panagiotis Louridas. Some guidelines for non-repudiation protocols. *SIGCOMM Comput. Commun. Rev*, 30(5):29–38, 2000.

[29] D. Moberg and R. Drummond. MIME-based secure peer-to-peer business data interchange using HTTP, Applicability Statement 2 (AS2). Technical Report RFC 4130, IETF, July 2005.

[30] T Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[31] Faisal Nabi. Secure business application logic for e-commerce systems. *Computers & Security*, 24(3):208–217, 2005.

[32] Henning Pagnia, Holger Vogt, and Felix C. Gärtner. Fair exchange. *Comput. J*, 46(1):55–75, 2003.

[33] Daniel M. Reeves, Benjamin N. Grosof, Michael Wellman, and Hoi Y. Chan. Toward a declarative language for negotiating executable contracts. Technical Report RC 21476 (96914), IBM, May 1999.

[34] Mark Richters and Martin Gogolla. Ocl: Syntax, semantics, and tools. In *Object Modeling with the OCL*, pages 42–68, 2002.

[35] Peter Ryan, Steven Schneider, et al. *Modelling and Analysis of Security Protocols*. Addison-Wesley, 2001.

[36] Al Farooq Salam, Lakshmi S. Iyer, Prashant Palvia, and Rahul Singh. Trust in e-commerce. *Commun. ACM*, 48(2):72–77, 2005.

[37] W. Scacchi. Process models in software engineering. In *Encyclopedia of Software Engineering, 2nd*, pages 993–1005. Wiley, 2002. Available on `http://citeseer.ist.psu.edu/scacchi02process.html`.

[38] Kathrin Schier. Multifunctional smart cards for electronic commerce-application of the role and task-based security model. In *ACSAC*, pages 147–154, 1998.

[39] S Schneider. Formal analysis of a non-repudiation protocol. In *Computer Security Foundations Workshop, 1998. Proceedings. 11th IEEE*, pages 54–65, Washington, DC, USA, 1998. IEEE Computer Society.

[40] Steve Schneider. Security properties and csp. In *IEEE Symposium on Security and Privacy*, pages 174–187, 1996.

[41] B-C Su. Risk behavior of internet shopping: comparison of college students' versus non-student adults'. In *ICEC '03: Proceedings of the 5th international conference on Electronic commerce*, pages 181–185, New York, NY, USA, 2003. ACM Press.

[42] Yao-Hua Tan and Walter Thoen. Formal aspects of a generic model of trust for electronic commerce. *Decision Support Systems*, 33(3):233–246, 2002.

[43] Theodosios Tsiakis and George Stephanides. The concept of security and trust in electronic payments. *Computers & Security*, 24(1):10–15, 2005.

[44] U.S. Census Bureau. Quarterly retail e-commerce sales - 4th quarter 2005, February 2006. Available on `http://www.census.gov/mrts/www/data/html/05Q4.html`.

[45] Joseph T Wells. *Corporate fraud handbook : prevention and detection*. Hoboken, N.J., 2004.

[46] Wikipedia. Modeling languages, June 2006. Available on `http://en.wikipedia.org/wiki/Modelling_languages`, last update:20/06/06.

[47] Daoxi Xiu and Zhaoyu Liu. A formal definition for trust in distributed systems. In *ISC*, pages 482–489, 2005.

[48] John Yesberg and Marie Henderson. Applications of trusted review to information security. In *ACISP*, pages 305–319, 2001.