WORKFLOW SIMULATION FOR INTERNATIONAL TRADE

Qiang Dong

Supervised by Professor Clark Thomborson

A thesis submitted in partial fulfilment of the requirements
for the degree of Master of Science in Computer Science,
The University of Auckland
February 2002.

ABSTRACT

Reengineering business processes is considered by many researchers to be indispensable in order to survive and prosper in today's competitive world. International trade has received some academic attention as an application of Business Process Reengineering. A major issue in international trade is due to problems pertaining to mutual-trust between trading parties without a prior trading relationship. Professor Lee advocates a new concept called electronic trade scenario to bridge the disparity. He uses Documentary Petri Nets as a representation language for the electronic trade scenarios.

This thesis, on one hand, experiences the Business Process Reengineering project through a solid example -- GT3, a simplified Documentary Credit Procedure in international trade. Some of the currently available techniques and technologies being used in Business Process Reengineering are explored, and are used to analyse our GT3 model. Then GT3 is modelled by a graphical representation language called Linear Documentary Petri Net (LDPN). We obtain LDPN by modifying Professor Lee's Documentary Petri Net. A simulator written in SIMSCRIPT II.5 is given to execute GT3 for the purpose of demonstrating workflows among the parties involved.

On the other hand, a partially finished software tool which can be used to augment the process of Business Process Reengineering for international trade is presented. With this tool, users can design a new trade scenario and make modifications on existing scenarios. Then, the corresponding workflows within that particular scenario are simulated automatically. This software tool is written in SIMSCRIPT II.5 and GEMA. Evaluation of our implementation shows that it provides full environment support for running simulation, it is correct in implementation and it is feasible to ease the design and modification of trade scenarios for international trade.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Professor Clark Thomborson for his remarkable guidance during my research and writing of this thesis. But for his words of inspiration, and encouragement, I could not have got this far. His rich experience and wealth of knowledge have greatly broadened my views on E-commerce and simulation. I have learnt much from him -- not only research techniques, but also programming and English language skills. This learning experience will go a long way in my professional/research career.

I would like to thank the Computer Science Subject-Librarian, Mrs. Hester Mountifield for her excellent guidance on literature survey, thanks to her expertise in electronic data retrieval.

I wish to thank my officemate Joshua Arulanandham for proofreading my document. I wish him all the best in his PhD research.

I also wish to thank Lloyd Tran of World Trade Centre, New Zealand for his regular participation in our discussions.

I wish to thank Ian Young, Rodney and Zoe Alderson for proofreading my thesis, and for offering me valuable advice.

I would like to thank my family for their everlasting support during my academic endeavours. It is their sharing, laugh, and complaints that give me the energy and motivation to complete this thesis.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1 Introduction

In this chapter, we discuss the main objectives and goals of our research, which is to build a training tool to be used in Business Process Reengineering. This tool can help the users to design simple trade scenarios. Also, users can modify the model, already built, to see the change in workflows associated with the changes made within the system. The organization of the thesis is given at the end of the chapter.

1.1 Background

Over the past decade, globalisation and the rapid advance of Information Technology have meant that enterprises face unprecedented changes. Against this backdrop, the concept of Business Process Reengineering (BPR) quickly caught the imagination of enterprise leaders and business process researchers. Fuelled by the continuing demand for enterprise transformation, there have been a lot of techniques, technologies and tools for conducting Business Process Reengineering projects.

Hammer (Hammer & Stanton, 1995) defined Business Process Reengineering as:

> The fundamental rethinking and radical redesign of business processes to bring dramatic improvements in performance.

According to Hammer (Hammer & Champy, 1992), business process is a set of activities taken together to produce a result of value to a customer. Reengineering business process is considered by many researchers to be a must in order to survive and prosper in today's competitive world.

International trade has received some academic attention as an application of Business Process Reengineering. Professor Lee (Lee, 1999) advocates a new concept called electronic trade scenario to bridge the gap between trading partners without previous trading relationships. He uses Documentary Petri Nets as a representation language for the electronic trade scenarios. A production-level case tool called InterProcs is

developed by his research team for the purpose of designing and prototyping new electronic trade scenarios.

1.2 Objectives of the Thesis

Our main objective is to develop a software tool to augment the process of Business Process Reengineering for international trade. With this tool, users can design a new trade scenario and make modifications on existing scenarios. Then, the corresponding workflows within that particular scenario are simulated automatically. Managers can experiment with this tool to help planning, decision-making, and predicting; users can also play with it to improve the understanding of the way the system works.

Our preliminary objective is to experiment with some of the currently available techniques and technologies to make us familiar with the Business Process Reengineering field and answer some of our questions such as: how to represent international trade with a model, how this model can be implemented in a simulation language, etc. Our contribution is a new model to simulate the workflows within a particular trade scenario in international trade.

1.3 Research Methodology

In order to achieve our objectives, we designed a trade scenario GT3 by simplifying the Documentary Credit Procedure in international trade, representing it with a graphical representation language. This representation language is obtained by modifying Professor Lee's Documentary Petri Nets. We call the resulting representation language Linear Documentary Petri Net which is more structured and easier to manage for the purpose of demonstration.

In our first implementation, we use a special purpose simulation language called SIMSCRIPT II.5 to program the simulator for our preliminary objective. We start from mapping our GT3 model to basic elements in SIMSCRIPT II.5 language, and finally produce an executable GT3 application for demonstrating our GT3 model.

In our second implementation, we use SIMSCRIPT II.5 and GEMA to program to achieve our main objective. We write pattern files in GEMA, which is a macro processor based on pattern matching. These pattern files can accept predefined structure statements written by user in an input file, and translate this input file into executable

SIMSCRIPT II.5 source code as an output file automatically. After the second implementation, we identify the advantages of programming in SIMSCRIPT II.5 and GEMA together for better understanding and efficient construction of business process models for international trade.

1.4 Organization of the Thesis

The various chapters in the thesis are organized as follows:

Chapter 2 presents a general overview of International trade, particularly Electronic Data Interchange and Documentary Credit Procedure.

Chapter 3 explores some of the currently available techniques and technologies being used in Business Process Reengineering projects for international trade.

Chapter 4 discusses the previous work done by Professor Lee: Documentary Petri Net and InterProcs. Then, the chapter describes our new graphical representation language called Linear Documentary Petri Nets, a modification of Professor Lee's Documentary Petri Nets. Lastly, the chapter describes a simple trade scenario GT3, and then represents it with Linear Documentary Petri Nets for further study.

Chapter 5 describes how simulation is done with GT3; we start with discussion on the choice of simulation model and implementation language and then evaluate the simulator.

Chapter 6 presents a review of the thesis, the summary and several possible areas that have potential for further work.

Chapter 2 Introduction to International Trade

According to Soloman (Soloman, 2000), it is widely acknowledged that enterprises that successfully develop an international market for their products generally increase their growth rate and profitability more rapidly than comparable companies trading locally. He also summarizes some advantages from international trade for enterprises.

The large markets that exist overseas offer a greater opportunity for the growth of a company. Overall product quality tends to be improved so as to satisfy the demands of the international marketplace. This also can have the effect of increasing domestic sales. Management abilities have improved with the need to understand and work within the international competition.

The need to come to terms with other cultures, marketing techniques and negotiating strategies can lead to the development of new skills and the refinement of existing ones. Also the increase in sales volume generated by international orders creates employment and job opportunities.

In this chapter, we first of all introduce the development of documentary process in international trade. Then we discuss the advantages of Electronic Document Interchange (EDI) over traditional or manual handling of paper business documents in international trade. Proliferation of EDI, as a fundamental infrastructure greatly enables the development of information technology and international trade. After that we introduce Documentary Credit Procedure. Finally, we summarize the chapter.

2.1 International Trade

Two centuries ago, the international trade community preferred the bills of lading as the documentary process to conduct the carriage of goods by sea between the parties involved. According to Kindred (Kindred, 1988), great changes have taken place in ocean transportation in the past fifty years; the shipping industry today operates in a

more reliable and efficient way due to the technological innovations. This greatly accelerates the cargo shipments.

An example given by Kindred (Kindred, 1988) shows that a container of goods is likely to reach the discharging port before its airmailed bill of lading on the North Atlantic route between Canada and Europe. So the carrier and the cargo owner are put to extra cost and delay waiting for the bill of lading. New forms of shipping contracts are needed to replace the increasingly inconvenient bill of lading.

In the 1970s, there were some documentary improvements; one of them is the documentary standardization. A big change in documentary design has been achieved by the creation of look-alike forms for common trade and shipping functions that are suitable for use anywhere in the world.

The number of documents and forms associated with each sale and movement of goods has been reduced sharply, so the international trade procedure is simplified. For instance, if a shipment of goods required thirty different forms, this shipment could be moved and controlled with only eight revised documents. However, Kindred (Kindred, 1988) thought that the most important documentary innovation for maritime transportation is the sea waybills.

What Kindred (Kindred, 1988) identifies as one of the unique features of the bill of lading is that, since it is the document of title to goods, it must be surrendered against delivery. The sea waybills looks like a blank-back bill of lading and acts similarly as a receipt for the goods and as a contract for their carriage, but it can't be negotiated as a document of title.

There is no need to present a sea waybill at destination in order to receive the goods. The customer has only to identify himself to the satisfaction of the carrier as the party, or his agent, named in the waybill in order to have the cargo released to him. In this way, no document is required at destination at all. Hence the use of sea waybills in place of bills of lading can expedite the delivery of goods by sea to the benefit of carrier and cargo owner as well.

From the 1980s, the new conditions of world trade have encouraged the shipping industry once more to develop novel shipping procedures, one of which is the

application of electronic data processing. It moves its fundamental premises from paper-based concepts represented by the ocean bill of lading into the modern electronic format. Thus the same information that would be contained in a bill of lading may in principle be expressed in electronic information and may be accessed by any authorized user with a connected and compatible terminal.

In this section, we introduced a bit of history on international trade, especially on the document improvement. Starting from 1980s, thanks to electronic data processing, international trade has evolved from paper-based format represented by the bill of lading to the modern electronic format. This leads the international trade into a new era. In the next section we will discuss Electronic Data Interchange.

2.2 Electronic Data Interchange

Although electronic commerce has been a subject of much media attention in recent years, the phenomenon is not new. Electronic Data Interchange (EDI) and Electronic Funds Transfer (EFT) associated with a monetary transaction has been around for a long time, like transferring salaries to employees' bank accounts and regular business partners making electronic commerce a routine part of daily business (Hoffman & Novak, 1997).

It is the transaction of business over the Internet that has attracted most attention, particular the e-tail version of e-commerce, because it opens up new channels for marketing and distribution. New business models have become possible with apparently low entry barriers (Delargy, 2001).

According to Sokol (Sokol, 1995), Electronic Data Interchange (EDI) is a method involving the exchange of transaction data between business partners in a standardized electronic format. Before Electronic Data Interchange became widely accepted, transactional data among businesses were represented in a variety of ways.

The EDI format evolved over time within the company, and became embedded within the old legacy systems and file processing techniques of the past. Alternatively, the company may have used any kind of software that provided its own internal methods of handling data. Regardless, there was little consistency in the way transactions between

trading partners were represented; each partner had to input data and handle their side of the transaction processing procedure separately.

This situation was considered by Sokol (Sokol, 1995) as a major bottleneck in maintaining trading relationships. Although data concerning the transaction was shared among the partners, re-entry and independent processing was required at each end. Electronic Data Interchange was created to eliminate such inefficiencies by establishing standards for the formatting of data concerning joint transactions. When such standards are employed by all trading partners, many shared processing activities can be facilitated.

Standard EDI documents are transferred between trading partners, and then automatically fed into the transaction processing systems. In some instances, the entire transaction, from initiation through to the final delivery of the goods or services, can be totally automated. In other situations, some human intervention is required to invoke manual procedures required to complete the transaction, but the standardized data format greatly improves the overall efficiency and quality of the process.

Among the most widely recognized advantage of Electronic Data Interchange is the speed in which transactions can be processed. This has furthered capabilities of organizations to adopt just-in-time logistics. Another value of Electronic Data Interchange is due to the ability to integrate the transaction handling processes with other computer based systems in the enterprise. This has reduced errors due to manual processing and provided better security. Finally, as a result of overall increase in efficiency, Electronic Data Interchange has provided cost savings.

The most important development in Electronic Data Interchange to dates regarded by Sophim (Sophim, 2001) is the creation of EDIFACT. In the mid-1970's, several Electronic Data Interchange standards began emerging in a number of countries led to the development of national standards. It became clear that if the Electronic Data Interchange standardization efforts were to meet the requirements of the international trade community, an Electronic Data Interchange standard for international trade was needed.

By the mid-1980's, the development of an international standard began taking shape within the United Nations. The standard is known today under the acronym

UN/EDIFACT, in full: United Nations Electronic Data Interchange for Administration, Commerce and Transport. In 1987, the syntax of UN/EDIFACT was approved as ISO Standard 9735.

The purpose is to develop recommendations and standards such as UN/EDIFACT to facilitate international trade. Over 50 countries and many international organizations, such as the European Commission, International Chamber of Commerce, and EAN International, are represented in UN/CEFACT. EDIFACT is now generally accepted as the international EDI standard that will be adopted by organizations that wish to trade on a global context.

Electronic Data Interchange standards for documents, software and network interfaces have eliminated the need for multiple proprietary solutions to handle data exchange between international trade partners. The global EDI user community defines the public standard for electronic business transactions, and they also establish open systems communications recommendations to be used with those standards.

An EDI system is based on three architectural elements as described by Sophim (Sophim, 2001):

- EDI Provider
- Application Software: in which the business information is created and used. Example is: order entry and accounts payable systems. For most of the companies, this application software is already in use.
- Translation Software: this is required to convert internal representation of data to and from the standard formats used in Electronic Data Interchange. This is usually supplied by the EDI provider in a package when subscribed. Such packages can be based on many types of equipment, from PCs to mainframes.

There is a great deal of flexibility in the way Electronic Data Interchange can be implemented. In the past, it has been relatively common for telecommunications companies to offer electronic mail facilities based on the OSI X.400 protocols. An X.400 email service had been used as a carrier for EDI messages for a long time.

According to Sophim (Sophim, 2001), EDI users can establish a leased line or dial-in link to the provider. The provider implements a “mail box” where EDI messages are lodged to wait for delivery to the recipient. There is obviously a problem where trading partners are connected to a different EDI service. EDI users have to subscribe to more than one provider.

Entrepreneurs are able to start new businesses more easily by accessing the World Wide Web. Engineers, product developers, and managers thousands of miles apart can collaborate to design and manufacture new products more efficiently; businesses can work more efficiently with their suppliers and customers. Consumers have greater choice and can shop in their homes for a wide variety of products from manufacturers and retailers all over the world, and they will be able to view these products, access information about the products, and order and pay for their choices, all from their home computers.

On legal issues arising in the use of Electronic Data Interchange, Kindred (Kindred, 1988) points out that it takes a great many years for the law to recognize the merits of new documentary systems and to give full endorsement to their use. The law had radically to revise how it treated dealings in goods in international trade. These electronic communications have made possible the transmission of trade and transport data without documents. It is to be hoped that the law can meet the challenge of the new shipping and trading environment. Particularly necessary is a legislative audit, country by country, with respect to the use of electronic documents. There are some positive signs that the law can be adapted, but greater and quicker reforms are necessary to complement the pace of change within the system of international trade.

Professor Lee (Lee, 1999) points out that although Electronic Commerce provided us a great advantage, many businesses and consumers are still wary of conducting extensive business over the Internet because of the lack of a predictable legal environment governing transactions. This is particularly true for international commercial activity where concerns about intellectual property protection, privacy, security, and other matters have caused businesses and consumers to be cautious.

Electronic Data Interchange is a way of business life, which is based on the principle of trust and contractual obligations. As Kindred (Kindred, 1988) points out, Electronic

Data Interchange cannot be introduced in a significant way unless we have a complete overhaul of a working system, methods and procedures. Above all unless the Laws/Acts governing business are amended to recognize EDI transactions, full-fledged EDI is not possible.

Once EDI transactions are recognized by Laws and Evidence act, and are provided for fast settlement of disputes, it should be possible to do away with requirements for paper documentations. That means there would be no necessity to submit invoices, packing list, etc in paper. Records need only be kept at the offices of Importers, Exporters for a minimum period of time for verification by concerned authorities in case needed.

In this section, we discussed Electronic Data Interchange. EDI is an automated method of placing electronic transactions. It reduces and simplifies paper based trading, advances the use of information technology throughout the trading cycle and develops and promotes paperless trading. The transactions involve transmitting standardized messages among trading partners.

Electronic Data Interchange has significant advantages over traditional or manual handling of paper business documents. EDI document processing occurs in seconds rather than the days or weeks it takes for manual processing. EDI is both faster and more accurate than manual processing, so the cost for administration and error correction drops significantly. Therefore, EDI users realize dramatic monetary and efficiency savings. In the next section we will discuss Documentary Credit Procedure.

2.3 Documentary Credit Procedure

Parties intending to conduct electronic commerce have to know about one another's "way of doing business" before they can begin exchange data electronically. Knowledge about each partner's preferred way of doing business must be conveyed to the other partner before the trade procedure. Otherwise, some misunderstanding will occur. An example of the battle of the forms is given by Professor Lee (Lee, 1999) below.

Consider only a simple post payment contract for goods. The buyer assumes that an invoice will be sent after delivery to trigger the payment obligation. The seller, on the other hand, abides by the practice that payment becomes due at the time of delivery,

and does not send an invoice. Thus, the goods arrive, but the buyer, waiting for an invoice, does not pay. The irked seller initiates collection proceedings.

Each party uses standardized documents such as purchase order, delivery agreement, etc. This indicates, typically on the backside in small print, the terms and conditions that are their style of doing business. Unfortunately the fine print is often ignored by the receiving party.

Documentary Credit Procedures were introduced to solve the problem we mentioned earlier: lack of trust and misunderstanding among trading partners in international trade. When partners don't know whether they can trust one another or not, the risks for both buyer and seller are very high. For example, the buyer may pay for the goods without being sure of receiving them; the seller may deliver the goods without being sure of receiving the payment.

For trade in a well-established industry area, standardized practice is generally accepted and there is usually no problem. These problems most probably happen in international trade, since a common legal and banking system exists only for trade conducted within the same country. In a Documentary Credit Procedure, the buyer and seller's bank take over the risks for them, so the buyer and seller rely on a trusted relationship between their banks.

Commonly, payments in international sales contracts are executed by Documentary Credits Procedure, subject to rules set forth in the Uniform Customs and Practice for Documentary Credits. In such transactions payment for the goods is made not on the delivery of the goods themselves but on the presentation of stipulated documents, which may include a commercial invoice, an insurance certificate, a certificate of origin and a transport document. The seller receives payment by presenting the stipulated documents to the bank that the buyer has instructed to make payment.

Here we presented a schematic of workflow in Documentary Credit Procedure in international trade shown in Figure 2.1. We redraw this figure from (Sophim, 2001). The numbers in the boxes correspond to the following stages of data interchange; the terms within the bracket are corresponding EDIFACT standardized messages.

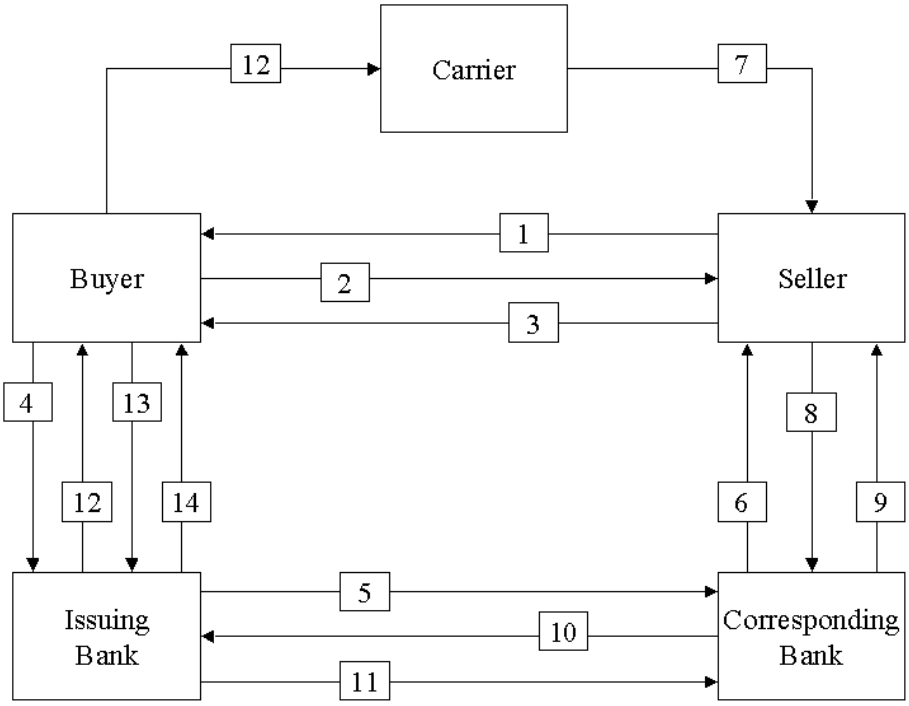


Figure 2.1 Simplified Workflow of the Documentary Credit Procedure.

1. Seller sends a price catalogue to buyer (PRICAT).
2. Buyer sends an order to seller (ORDER).
3. Seller sends an invoice to buyer (INVOIC).
4. Buyer sends a letter of credit request to issuing bank (DOCAPP).
5. Issuing bank sends a letter of credit to corresponding bank (DOCADV).
6. Corresponding bank sends a letter of credit to seller (DOCADV).
7. Carrier sends a bill of lading to seller (IFTMAN).
8. Seller sends a bill of lading to corresponding bank (IFTMAN).
9. Corresponding bank sends a credit advice to seller (CREADV).
10. Corresponding bank sends a bill of lading to issuing bank (IFTMAN).
11. Issuing bank does payment to corresponding bank (FINPAY).
12. Issuing bank sends a bill of lading to buyer (IFTMAN).
13. Buyer sends a payment order to issuing bank (PAYORD).
14. Issuing bank sends debit advice to buyer (DEBADV).

2.4 Summary

From the paper-based bill of lading to the modern electronic-formatted standardized UN/EDIFACT message, international trade has changed greatly during the past several decades. Electronic Data Interchange is a means of paperless trading and was a natural

evolution from paper documents as data carriers, to computer and telecommunication systems as automatic carriers and processors of data.

In traditional business processes, such as ordering and invoicing, paper documents contain structured information in various boxes. In Electronic Data Interchange, this information is mapped into a structured electronic message. In operation, Electronic Data Interchange is the interchange of these agreed messages between trading partners to ensure speed and certainty and better business practice in the supply chain.

Sophim (Sophim, 2001) states that standards are of the utmost importance to EDI. Without pre-defined agreed standards EDI is of little value. By using standard messages, organizations can exchange information between many different trading partners and be sure of common understanding.

Documentary Credit Procedure has been developed as a compromise between the seller's need for the security of ownership of the goods and speed of payment, and the buyer's need for speed of transit but the longest period for payment. The international banking system and the Documentary Credit Procedure thus provide both with security that they need.

Since the Documentary Credit Procedure affords buyer and seller equal protection it has evolved as the method preferred for first-time dealings when buyer and seller are unknown to each other. The seller is unsure of the buyer's ability, or intention, to make payment while the buyer is unsure that the seller will deliver the correct goods in a timely fashion.

Chapter 3 Business Process Reengineering

Over the past decade, firms have faced unprecedented change: globalisation, political realignments, and the rapid advance of Information Technology. Against this backdrop, the concept of Business Process Improvement quickly caught the imagination of corporate leaders. Early success stories pushed Information System executives to take an active role in Business Process Improvement projects. In this chapter, we introduce the concept of Business Process Improvement, also techniques and technology being used in Business Process Reengineering.

3.1 Business Process

If you have ever waited in a queue in a grocery store for a long time, you can appreciate the need for process improvement. In this case, process refers to the checkout process, and the purpose of the process is to pay for and pack your groceries. The process begins with you stepping into line, and ends with you receiving your receipt and leaving the store. You are the customer, and the store is the supplier.

In this example, what we have described is to be considered a business process. The steps involved in this process are the activities that you and the store personnel do to complete the transaction. We can imagine other business processes: ordering clothes from mail order companies, requesting new telephone service from Telecom, etc.

There is no clear and agreed definition of a business process available in the literature.

Business process is defined by Hammer (Hammer & Champy, 1992) as:

> A set of activities that, taken together, produces a result of value to a customer.

According to Davenport (Davenport, 1993), business process is:

> An ordering of work activities across and place, with a beginning, and an end, and clearly identified inputs and outputs.

Earl (Earl, 1994) defines business process as:

> A lateral or horizontal form, which encapsulates the interdependence of tasks, roles, people, departments and functions required to provide a customer with a product or service.

On the other hand, Saxena's (Saxena, 1996) definition of business process is:

> A set of inter-related work activities characterized by specific inputs and value added tasks that produce specific outputs.

There is no consensus amongst the authors on defining the business process. From the above, we can identify some common elements in the majority of those definitions.

As Hlupic (Hlupic & Robinson, 1998) states, these elements relate to the process itself, which is usually described as transformation of input, or a set of activities; process input; and process output which is usually related to creating value for a customer, or achieving a specific goal.

To remain competitive in today's global economy, companies are being forced to re-evaluate the way they do business with their customers and their vendors. At the same time, there is a growing emphasis on flexibility, in being able to respond quickly to changes in consumer preference and demand.

Besides those two factors mentioned above, the necessity of delivering high quality products to gain and maintain customer loyalty and the necessity of rigidly controlling and reducing costs, define the challenge of business facing by enterprises in the future. In the next section we will introduce the models of Business Process Improvement to get an idea on how business process can be improved.

3.2 Models for Business Process Improvement

Over the last 10 to 15 years, companies have been forced to improve their business processes because customers are demanding better products and services. If the customers cannot receive what they want from one supplier, they have many other suppliers to choose from.

3.2.1 Continuous Process Improvement (CPI)

Many companies began Business Process Improvement with a continuous improvement model. Kaizen (Sharp & McDermott, 2001) is the concept of Continuous Process Improvement in Japan. Continuous improvement is also called total quality management (TQM) by Sharp et al (Sharp & McDermott, 2001). In this thesis, we will use Continuous Process Improvement instead of Kaizen or TQM to preserve consistency. The central principle of Continuous Process Improvement is that business processes need to be improved continuously to keep a quality product in production.

Continuous Process Improvement (CPI) as shown in Figure 3.1, starts by understanding and measuring the current process, and then makes performance improvements accordingly. This model begins by documenting what the current process, measures the process based on what customers want, executes the process, measures the results, identifies improvement opportunities based on the data being collected, implements the process improvements, and measures the performance of the new process.

This loop repeats over and over again, and is called Continuous Process Improvement. Figure3.1 illustrates the basic steps in Continuous Process Improvement. We adapt this figure from (Hiatt, 2001) and redraw it.

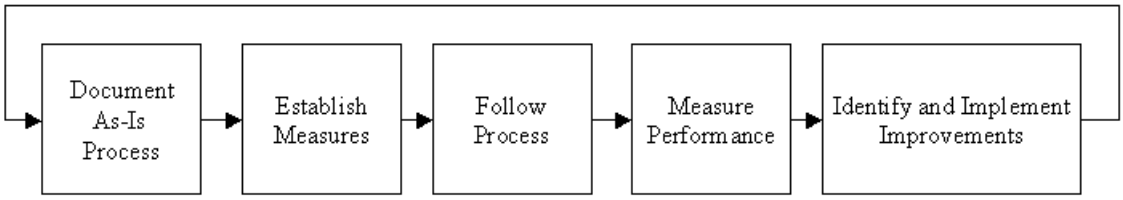


Figure 3.1 Continuous Process Improvement Model.

Continuous Process Improvement for improving business processes is effective to obtain gradual, incremental improvement. However, over the last decade several factors have accelerated the need to improve business processes. New technologies are rapidly bringing new capabilities to businesses. So business processes can be improved dramatically. Another apparent trend is the opening of world markets and increased free trade. Such changes bring more companies into the marketplace, and competing becomes harder and harder.

As a result, companies have required methods for faster Business Process Improvement. Moreover, companies want breakthrough performance changes, not just incremental changes. Because the rate of change has increased for everyone, few businesses can afford a slow change process. One approach for rapid change and dramatic improvement that has emerged is Business Process Reengineering (BPR). This will be introduced in the next part of this section.

3.2.2 Business Process Reengineering

According to Davenport (Davenport & Short, 1990), Business Process Reengineering is:

> The analysis and design of workflows and processes within and between organizations.

Teng et al (Teng, Jeong, Kettinger, & Grover, 1995) define Business Process Reengineering as:

> The critical analysis and radical redesign of existing business processes to achieve breakthrough improvements in performance measures.

Hammer (Hammer & Stanton, 1995) defined Business Process Reengineering as:

> The fundamental rethinking and radical redesign of business processes to bring dramatic improvements in performance.

Hammer also gave detailed explanations of the four key words in his definition: dramatic, radical, process and redesign. We summarize his explanations below.

With the concept of “dramatic” improvement, Business Process Reengineering is not used to make marginal improvements, such as making the current process 5 percent or 10 percent better. It is used to make “quantum leaps in performance, achieving breakthroughs.” Improvements, such as cost reduction or speed increases, can be measured on the intention of the organizations.

The second key word is “radical” which means going to the root of things. Business Process Reengineering is not about improving what already exists; instead, it is about starting from the beginning and reinventing the business process.

The third key word in the definition is “process”. Hammer regarded a process a group of related tasks that together create value for a customer. An example given by Hammer (Hammer & Champy, 1992) is shown as follows:

> Order fulfilment is a process, comprising a series of tasks: receiving the order, entering into a computer, checking the customer's credit, allocating inventory from stock, picking the inventory out of the warehouse, packing it into a box, loading the box into the truck and so on.

For the customer, the only concern is the end result -- the delivered goods that is created by the sum of all these related activities, not each separate activity. So the processes are at the very heart of every enterprise. They are the means by which companies create value for their customers.

The fourth key word in Hammer’s definition is “redesign”. Business Process Reengineering is about the design of how work is done. Usually when we think of design, it is applied to products. But here, when talking about reengineering, it is based on the premise that the design of processes is of essential importance. Maybe the employees are smart and capable, well trained, highly motivated, but if the work they are doing is poorly designed, it will not be well executed. The starting point for organizational success is well-designed processes.

Business Process Reengineering assumes the current process is irrelevant and starts over. Such a perspective enables the designers to separate themselves from the current process, and focus on a brand new process. Business Process Reengineering is illustrated by Figure 3.2. We redraw this figure from (Hiatt, 2001).

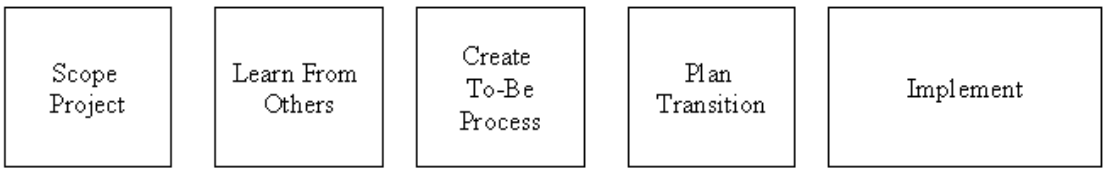


Figure 3.2 Business Process Reengineering Model.

Figure 3.2 shows Business Process Reengineering begins with defining the scope and objectives of the reengineering project, then going through a learning process. We can learn from customers, employees, competitors, and make full use of new technology.

Given this knowledge base, a vision for the future can be created and new business processes could be designed. It is then a matter of implementing the resulting solution.

The difference between Continuous Process Improvement (CPI) and Business Process Reengineering (BPR) lies in where it starts, that is, whether it starts with current process or with a clean slate; and also with the magnitude and rate of resulting changes.

When BPR first emerged, there was a certain tension between Business Process Reengineering and Continuous Process Improvement communities. Reengineers wondered why their CPI counterparts were improving processes that ought to have been thrown away. And those CPI engineers found the reengineers to be rushed and ruinous.

3.2.3 Process Management

Over the last few years, the concept of Business Process Reengineering and Continuous Process Improvement has emerged. Now Business Process Reengineering and Continuous Process Improvement have been combined under the milder term Process Management. Sharp et al (Sharp & McDermott, 2001) explain their Process Management model using Figure 3.3. We redraw this figure from (Sharp & McDermott, 2001).

This emerging effort attempts to address the difficulties of implementing major change in enterprises. It makes Business Process Reengineering a broader, yet more comprehensive process management concept (Davenport & Stoddard, 1994).

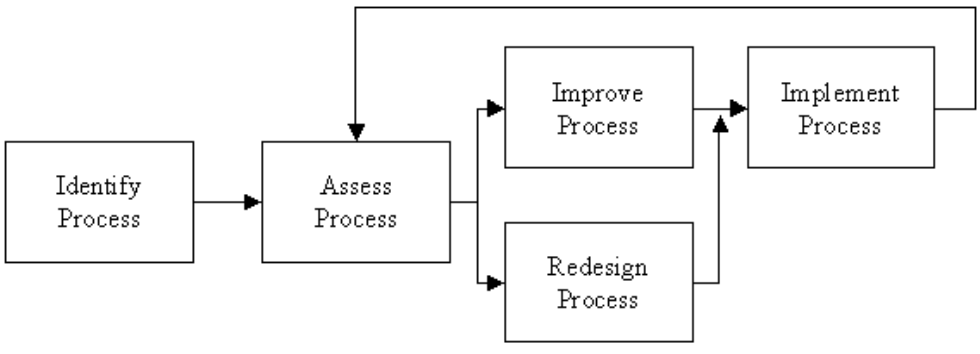


Figure 3.3 Process Management Model.

Process Management starts with identifying the processes understudy, then measures the process based on what customers want. According to the assessing results, decision can be made on whether to improve or redesign the process. Then implementation can

be done after that. The most important character of this Process Management is that there is a loop in the Process Management. The performance of the business process can be improved by repeating those steps: assessing, improving or redesigning, and implementing. These steps are shown in Figure 3.3.

Sharp et al (Sharp & McDermott, 2001) state that it is very difficult to find whether Business Process Reengineering or Continuous Process Improvement can exactly match the particular requirements of a company. The challenges are to know what method to use, when, and how to make it successful.

In this section we introduced the concept of business process and Business Process Improvement. According to Hammer (Hammer & Champy, 1992), business process is a set of activities that produces a result of value to a customer. Due to the globalisation and rapid advance of information technology, Business Processes Improvement are demanded by many companies in order to provide better products and services for customers.

There are three models in Business Processes Improvement, Continuous Process Improvement (CPI), Business Process Reengineering (BPR), and Process Management. Hammer defines (Hammer & Champy, 1992) Business Process Reengineering as the fundamental rethinking and radical redesign of business processes so as to improve the performance dramatically. Continuous Process Improvement improves business process performance by gradual, incremental improvement. Process Management is the combination of the previous two models.

The radical redesign characteristics can make dramatically improvement and achieve breakthroughs in business process performance, so we concentrate on Business Process Reengineering in later discussion in the rest of the thesis.

A lot of leading organizations have conducted Business Process Reengineering for improving productivity and gaining competitive advantage. But unfortunately, a survey by Hammer and Champy (Hammer & Champy, 1993) suggest that the failure rate can be as high as 70%. The figure is supported by a number of consulting studies both in North America and Europe.

Due to the high failure rate in many real-life business change projects, Business Process Reengineering success factors have become an important area of study. In the next section we will briefly discuss some of the success factors with Business Process Reengineering.

3.3 Success Factors for Business Process Reengineering

A survey by Jackson (Jackson, 1996) in 1996 shows that during the past few years 180 US and 100 European companies found that 75% of these companies had engaged in significant reengineering efforts. Over a 24-month period, a benchmarking study (Hiatt, 2001) was made with more than 150 companies. The resulting success factors that can lead to successful outcomes for Business Process Reengineering projects from this benchmarking study are: Top Management Sponsorship, Strategic Alignment, Compelling Business Case for Change, Proven Methodology, Effective Change Management, Line Ownership, Reengineering Team Composition.

After carefully examining these seven factors, we found that nearly all of them are related to human resources management; only Proven Methodology is related to technical side. When we say "technical side", we mean the application of engineering principles in business process design and analysis.

Hlupic (Hlupic & Robinson, 1998) states that some of the frequently mentioned problems related to Business Process Reengineering include the inability to accurately predict the outcome of radical change, difficulty in capturing existing processes in a structured way, shortage of creativity in process redesign, the level of costs incurred by implementing the new process, or inability to recognize the dynamic nature of the processes.

Kettinger et al (Kettinger, Teng, & Guha, 1997) point out that the lack of a comprehensive, scientifically grounded design methodology to structure, guide, and improve organizational design efforts results in the high failure rate in Business Process Reengineering projects.

Meel and Sol (Meel & Sol, 1996) advocate the development of computer-based models of business process as a crucial mechanism to support the process of experimentation

with alternative business structure so as to decrease the failure rate in Business Process Reengineering projects.

From the above discussion we can see that to make a Business Process Reengineering project successful, we need to put efforts on both human resource management side and technical side as well. In this thesis, we will focus on the technical side of Business Process Reengineering.

As we have discussed before, there are three models for Business Process Improvement. They are Continuous Process Improvement, Business Process Reengineering, and Process Management. Although each model in Business Process Improvement differs in the scope and range of the anticipated changes in business process and the business context in which they can be used, however they all have in common that they require businesses to model the ways in which they currently operate, to identify opportunities for change, and to design and implement alternative ways of carrying out business processes.

In the view of the above, Business Process Modelling (BPM) has recently received widespread attention and has been acknowledged as an integral part of any change management project according to Swami (Swami, 1995).

Willcocks et al (Willcocks & Smith, 1995) and Galliers (Galliers, 1993) also state that businesses and business processes are sufficiently complex systems and therefore carefully developed models are necessary to understand their behaviour in order to be able to design new systems or improve the operation of existing ones. In the next section, we will discuss the techniques for Business Process Modelling.

3.4 Techniques for Modelling Business Process

3.4.1 Business Process Reengineering and Information System

The role of Information Technology as an agent for organizational change has been heavily emphasized in the literature. Information System (IS) has usually become the major vehicle on which business change relies. Indeed, the idea of aligning the design of organizational processes with the associated Information Technology infrastructure has been one of the major driving forces for the development of change management paradigms like Business Process Reengineering (Davenport, 1993).

Reengineering a business process is a very important and crucial task in order to step from the old-fashioned Information Systems to a new generation of Information Systems. Such new systems integrate large amount of information services, which are distributed over a wide computer network and support cooperation and collaboration among them. They are able to integrate legacy and new system components, and are customized, and reflect the high dynamics of today's business process.

Davenport and Short (Davenport & Short, 1990) argue that Business Process Reengineering requires taking a broader view of both Information Technology and business activity, and of the relationships between them. Information Technology should be viewed as more than an automating or mechanizing force: to fundamentally reshape the way business is done.

Curtis et al (Curtis et al., 1992) indicate that the traditional modelling of Information Systems has focused on analysing data flows and transformations. This kind of modelling accounted only for the organization's data and that portion of its processes that interacted with data.

Newer uses of Information Technology extend computer use beyond transaction processing into communication and coordination. Successfully integrating these systems into the enterprise often requires modelling even the manual organizational processes into which these systems intervene.

3.4.2 Business Process Modelling Objectives, Goals and Requirements

Warren (Warren, 1996) states that the close attention being paid to business change management paradigms, such as Business Process Reengineering has created a market for process modelling techniques. Business Process Modelling has emerged as an important research and application area within organizational or inter-organizational design.

Business process models can be used to serve a wide number of applications, for example, to drive a strategic organizational analysis, to drive requirements and specifications for Information Systems design, or to support automated execution of process.

Business Process Reengineering is one of the applications that share a growing requirement to represent the processes through which work is accomplished. Process representation becomes a vital issue in redesigning work and allocating responsibilities between human and computers.

Curtis et al (Curtis et al., 1992) identify some basic uses for process models among a wide arrange of process modelling objectives. They are ranging from understanding aids to automated execution support of process modelling. These objects are: Facilitate human understanding and communication, Support process improvement, Support process management, and Automate execution support.

The goals and objectives of a particular study impact the use of the model, therefore influence the requirements posed on the process representation model. As a result, the context and objectives of a particular study make some Business Process Reengineering methods more suitable than others. The design or choice of a particular process modelling method needs to be chosen according to the intended use of the model.

Curtis' idea on the different Business Process Modelling goals and the typical Business Process Modelling requirements can be illustrated in Table 3.1. For example, in order to facilitate human understanding and communication on the business process under study, a model of the business process should be complete, that is, including all related parts of the interested process; the modelling should be comprehensive for better understanding of its users without much specific knowledge; and the model should be easy for the users to communicate with each other.

To support process improvement, Business Process Modelling should provide the ability to measure the performance of the system and to compare the performance of current system with that of other similar systems to support decision-making, etc.

3.4.3 Business Process Modelling Techniques

To satisfy the requirements of all possible users, a process model should be able to integrate and represent many forms of information. Because of that many requirements talked above, to set up successful process models would be quite difficult and could be result in very complex models, which could be very hard to use due to the complexity.

To deal with this complexity problem, a wide variety of process modelling techniques and approaches has been proposed. Almost all of these techniques are targeted to a limited subset of organizational modelling projects. They can provide constructs suitable for satisfying the requirements associated with only particular modelling goals as we discussed above. In this section we will give a brief overview of some of these techniques.

Curtis et al (Curtis et al., 1992) give a definition on model and state the purpose of the model as follows:

> An abstract representation of reality that excludes much of the world's infinite details. The purpose of a model is to reduce the complexity of understanding or interacting with a phenomenon by eliminating the detail that does not influence its relevant behaviour. Therefore, a model reveals what its creator believes is important in understanding or predicting the phenomena modelled…

Process Modelling Objectives and Goals	**Business Process Modelling Requirements**
Facilitate Human Understanding and Communication	Comprehensibility Communicability Completeness
Support Process Improvement	Measurability Comparability Decision Support Component Identification
Support Process Management	Forecasting Support Monitoring and Co-ordination Support
Automated Execution Support	Co-operative Work Support Automated Performance Measurement Support Process Integrity Check Support

Table 3.1 BPM Objectives, Goals and Requirements.

- Flowchart

A Flowchart is a formalized graphic representation of a program logic sequence, work or manufacturing process, organization chart, or similar formalized structure (Whatis, 2001).

> In computer programming, flowcharts were formerly used to describe each processing path in a program (the main program and various subroutines that could be branched to). Programmers were admonished to always flowchart their logic rather than carry it in their heads. With the advent of object-oriented programming and visual development tools, the traditional program flowchart is much less frequently seen. However, there are new flowcharts that can be used for the data or class modelling that is used in object-oriented programming.

Traditional program flowcharting involves the use of simple geometric symbols to represent the beginning or end of a program, a process, a decision, or an I/O process. Here we use an example, to buy a heavy-duty stapler for the office in a University, to illustrate how a flowchart works as a process modelling techniques.

First of all a requisition form must be filled out by the faculty member who wants to buy this stapler. Secondly, department head's signature must be acquired to keep on with this process. Thirdly, the completed form must be sent to the Business Office. If funds are available for this purchase, the Business Office can make a purchase order to the vender. Finally the faculty member just needs to wait for the heavy-duty stapler to show up. This process model in Flowchart is illustrated in Figure 3.4.

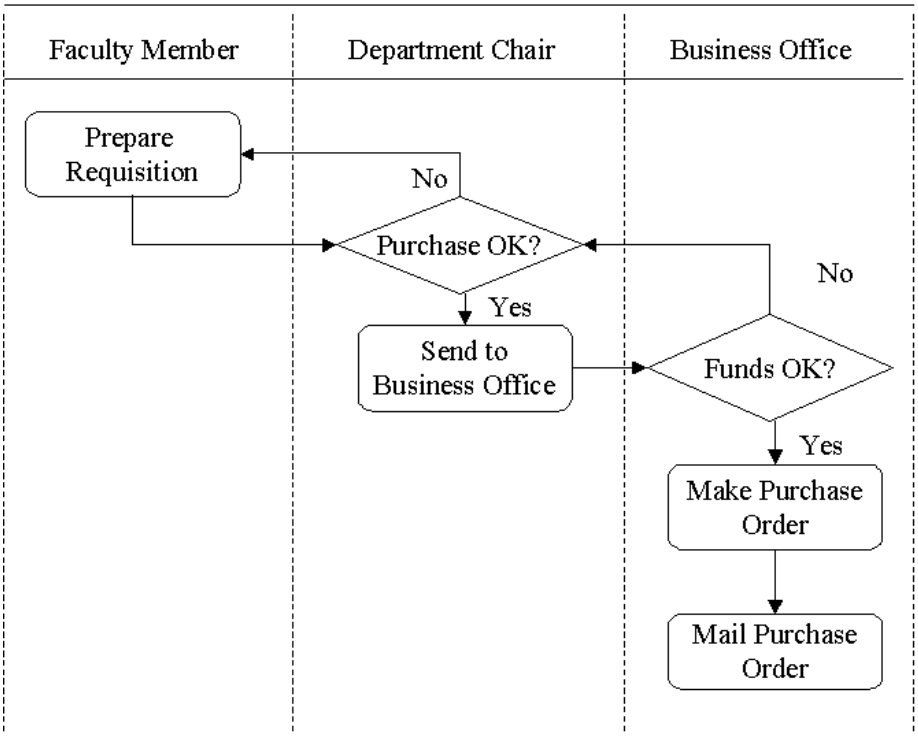


Figure 3.4 Heavy-Duty Stapler Purchase Process in Flowchart.

Meanwhile the Business Office is engaged in other series of steps such as checking for funds available, preparing the purchase order and placing the order. The supplier, in shipping the stapler, and the transporter, in delivering it, also engage in processes. If the stapler does not show up, something goes wrong in one of the processes. To find the lost stapler, we need to go back and check each step of the processes to identify where the stapler got lost.

Flowcharts provide a visual picture of a process. It is obvious and easy for people to understand the process being modelled. People can identify each step to be taken and each decision to be made. By working on a flowchart together, all the team members develop a shared understanding of the process and problems within the process.

Flowchart sounds like a simple task; however constructing Flowcharts accurately may take much more time than anticipated. Processes usually end up being more complicated than at first suspected (Kettinger, Teng, & Guha, 1997).

- Scenario Analysis

Clemons (Clemons, 1995) advocates the use of Scenario Analysis to manage the strategic risks associated with Business Process Reengineering. Scenario Analysis acknowledges the fact that the future is uncertain. The future discontinuities mean that companies cannot base their future strategies on the existing situation.

Instead, a range of potential futures should be identified and appropriate actions should be planned so that the company knows what operations to implement within each possible future.

However this scenario analysis is more or less a conceptual approach to Business Process Modelling instead of a representational modelling technique. No specifying process models are given by the author.

- System Dynamics

System Dynamics (SDS, 2001) is a methodology for studying and managing complex feedback systems, such as one finds in business and other social systems. Feedback refers to the situation of X affecting Y and Y in turn affecting X perhaps through a chain of causes and effects. In a system with feedback, we cannot study the link

between X and Y, and the link between Y and X independently to predict how the system will behave. Only the study of the whole system will lead to correct results.

- Knowledge-based Techniques

Compatangelo et al (Compatangelo & Rumolo, 1997) take a different approach and advocate the use of Knowledge-based techniques, with emphasis on automated reasoning, to address enterprise modelling at the conceptual level.

The authors indicated that knowledge based representation of enterprise models may be a valid mechanism for representing knowledge and deductive argumentation about business systems. Disadvantages of this approach are also obvious.

Knowledge-based techniques can represent only a partial view of an organizational system and existing tools do not possess the necessary depth and breadth of reasoning to be of practical value in reasoning about complex, real-life enterprise models.

Furthermore, knowledge-based techniques can accomplish only symbolic representation of processes. Therefore, they cannot accommodate requirements for process measurement and comparison that are integral to process improvement applications.

- Message Sequence Chart

According to Mauw (Mauw & Reniers, 1994), the Message Sequence Chart is used for visualizing systems, which run within communication systems. Message Sequence Chart can be viewed as a special trace language, which mainly concentrates on sending and receiving of messages among communicating processes. A Message Sequence Chart is not a description of the complete behaviour of a system; it expresses one execution trace. A collection of Message Sequence Charts may be used to give a more detailed specification of a system.

A basic Message Sequence Chart contains a description of the communication behaviour of a number of instances. An instance is an abstract entity of which one can observe the interaction with other instances or with the environment.

The Message Sequence Chart in Figure 3.5 defines the communication behaviour between instances i_1, i_2, i_3 and i_4. We redraw this figure from (Mauw & Reniers, 1994). An instance is denoted by a vertical axis. The time along each axis runs from top to bottom. A communication between two instances is represented by an arrow, which starts at the sending instance and ends at the receiving instance. In the above figure we consider the messages m_1, m_2, m_3 and m_4.

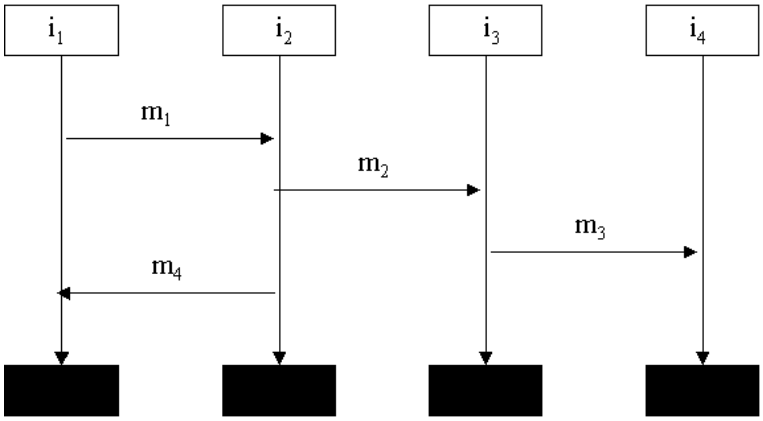


Figure 3.5 A Message Sequence Chart.

Although the activities along one single instance axis are completely ordered, we will not assume a notion of global time. The only dependencies between the timing of the instances come from the restriction that a message must have been sent before it is received.

In Figure 3.5 this implies for example that message m_3 is received by i_4 only after it has been sent by i_3, and, consequently, after the reception of m_2 by i_3. Thus m_1 and m_3 are ordered in time, while for m_4 and m_3 no order is specified.

A Message Sequence Chart contains the description of the communication between instances. For brevity, we restrict ourselves to the core language of Message Sequence Charts. So our Message Sequence Chart concentrates on communications only.

- Multilevel Modified Petri Net

Tsalgatidou et al (Tsalgatidou, Louridas, Fesakis, & Schizas, 1996) propose the use of Multilevel Modified Petri Nets (MPN) as an appropriate representation for the development of Business Process Models. They define a structure for MPNs, discuss

why this approach is appropriate for representing Business Process Models, and demonstrate their feasibility and usefulness in a real life case study.

They advocate that MPN enables modelling activities, resources, control, dataflow, and the organizational structure. The authors identify the following advantages of their method: firstly, model decomposition that allows for handling real-world complexity and facilitates better validation of the experimentation; secondly, ability to simulate and animate models that facilitates communication and decision-making.

- Swimlane Diagrams

A Swimlane diagram (Sharp & McDermott, 2001) can show an entire business process from the beginning to the end. Also it can be used both to understand the as-is workflow, and to design the to-be workflow. We explain the Swimlane technique in more detail than other techniques discussed in this section, because we will use this technique in Chapter 4 of this thesis.

Swimlane is popular (Sharp & McDermott, 2001) because it can highlight the relevant variables -- what is done, by whom, and in what sequence -- in a simple notation that requires little or no training for users to understand the business process under study. It can show a process at any level, from a very high level down to one showing each individual task.

The authors describe a workflow model as depicting the three Rs:

- Roles: the actors or process performers who participate in the process.
- Responsibilities: the individual tasks that each actor is responsible for.
- Routes: the workflows and decision that connect the tasks together, and therefore define the path that an individual work item will take through the process.

Here we use an example from Sharp et al (Sharp & McDermott, 2001) to illustrate the powerful visual effects provided by Swimlane Diagrams. This example models an enrolment process in a University shown in Figure 3.6. We redraw this figure from (Sharp & McDermott, 2001).

The actors in the process are listed down the left side of the diagram; each is given a Swimlane that extends to the right across the page, and delineated by dotted lines. Each task is represented by a box placed in the Swimlane of the actor that executes it. Arrows indicate the sequence and flow of tasks. A flow from an actor to another, that is the one crosses the line between Swimlanes, is called a handoff. Example of handoff, step, actor, and flow are indicated by a shadowed box in Figure 3.6.

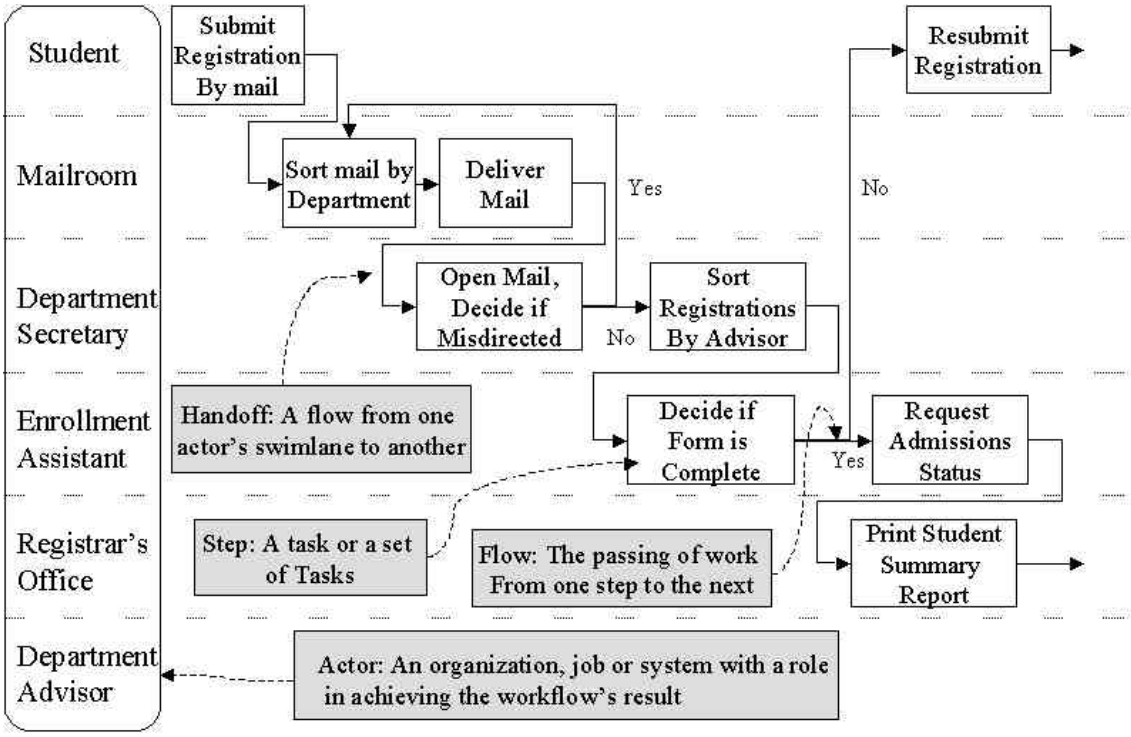


Figure 3.6 An Enrolment Process Represented by a Swimlane Diagram.

A Student submits a registration form to the University by mail; later when the mails are received in the Mailroom, they are sorted, and then delivered to the intending departments. At this stage the Department Secretary can open the mails. If the mail is misdirected, it will be sent back to the mailroom to be redirected.

Correctly directed mail is sorted by advisor delivered to the Enrolment Assistant; the Enrolment Assistant will decide whether the registration form is complete or not, if no, other process will applied to inform the student to complete the registration form; otherwise the complete registration form will be sent to the Registrar's Office ask for admission.

At the final stage of this specific process, the Registrar's Office will evaluate the application of this student and make a summary report; this report is handled by other process, such as sending mail back to the Student to inform the summary report.

There is no practical difference between an "actor" and a "role" for our research purpose, so these two terms are used interchangeably in this thesis. Generally, an actor is any identifiable person or group that handles the work between the initial event and the achievement of the process's result.

Sometimes a single person can perform multiple roles in a process. In this case, each role should be given a separate Swimlane initially. If subsequent analysis shows that the handoffs from one role to another occur seamlessly without delay, then these two Swimlanes can be collapsed into one. However, if the handoffs proved to be a source of delay, error, or expense, then the Swimlanes should be kept as they were. Guidelines for actor are illustrated in Figure 3.7. We redraw this figure from (Sharp & McDermott, 2001).

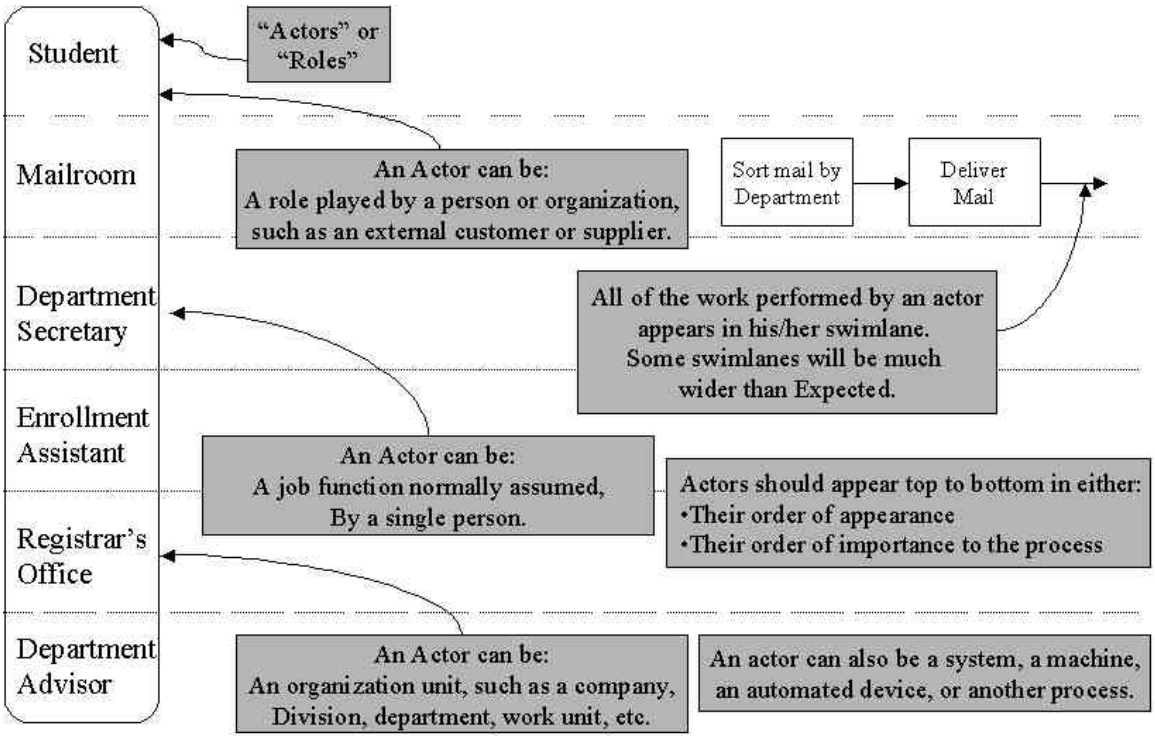


Figure 3.7 Guidelines for Actors.

While modelling a process using workflow, one difficulty maybe be encountered is deciding which steps to include in the workflow model. An actor may perform many

activities, but some of them may not belong to the process being studied. Sharp et al (Sharp & McDermott, 2001) use an example for analysing to draw out the guideline on what steps should be included in the workflow model.

In an insurance company, 5% of incoming claims were routed to an internal audit function and then returned to the normal claims flow. Whatever happened in the audit step never made any difference to the subsequent handling of the claims, so some analysts argued that it wasn't part of the process. But the problem was subsequent steps in resolving the claim could not proceed until a claim returned from the audit step, even though the audit step didn't actually do anything to the claim. So this step is a part of the process being studied and should be added in the workflow model.

Sharp et al (Sharp & McDermott, 2001) summarize the guideline for which steps to include as: "Show every step that adds value; moves the work along; or introduces delay."

"Adds value," means cause a state change in the direction of completion. The value could be subtracted as well; it will cause the state change in the opposite direction of completion.

"Moves the work along," indicates that some steps may not change the work item in any way, except to transport it between other steps in the workflow. Internal or external delivery services that handle work items belong to this category. These steps are important to the process behaviour, so they should be added in the workflow model.

"Introduces delay," reveals that a step may not change the state of the work item, or move it along, but some subsequent step cannot proceed until the delay-introducing step completes. The internal audit step in the claim-handling example is a good illustration of this category.

3.4.4 Summary and Discussion

In this section, we discussed some techniques currently being used in Business Process Modelling. They are Flowchart, Scenario Analysis, System Dynamics, Knowledge-based Techniques, Message Sequence Chart, and Swimlane Diagram. These techniques can provide constructs suitable for satisfying the requirements associated with only particular modelling goals.

Flowchart is specialized in graphic representation of a program logic sequence, work or manufacturing process. The disadvantage of Flowchart is that it is inadequate in modelling complex systems. System Dynamics is a methodology for studying and managing complex feedback systems only. We can only expect the right results when applying System Dynamics to a feedback system because System Dynamic lacks of ability to analysis and predict an independent participant. Scenario Analysis is a conceptual approach instead of a practical one to Business Process Modelling.

Knowledge-based techniques can accomplish only symbolic representation of processes. They cannot accommodate requirements for process measurement and comparison that are integral to process improvement applications. Message Sequence Chart expresses only one execution trace each time. It helps us to concentrate on the order of communication among parties. To describe the complete behaviour of a system, we need a collection of Message Sequence Charts to represent a more detailed specification of a system. This may bring confusion in visualization. Swimlane Diagram aims at the automation of business process. It gives us an overview on how the whole system works at any level.

Petri net has been proved to be useful in the context of logistics and production control by Van der Aalst (Van der Aalst, 1994). The author identifies that it is not restricted to logistics and manufacturing. Petri nets can also be used to support Business Process Reengineering efforts (Van der Aalst & Heea, 1996).

We get high-level Petri nets by adding extensions to classic Petri nets. These extensions allow for the representation and study of complex business process. In section 3.5 we will introduce Petri nets in detail.

Compared with Petri net, those techniques we mentioned above suffer from two important drawbacks: (1) the lack of formal semantics and (2) the absence of powerful analysis methods and tools. High-level Petri nets have formal semantics. A Petri net model of a business process is precise and unambiguous description of the behaviours of the system under study according to (Van der Aalst & Heea, 1996).

Despite the formal background, Petri nets are easy to understand. The graphical nature can be used to visualize business process in a natural manner and supports the communication among people involved in a Business Process Reengineering project.

Business Process Modelling Objectives and Goals	**What does it mean**	**Business Process Modelling Techniques**
Facilitate Human Understanding and Communication	1. Enable communication and agreement on the processes. 2. Formalize the process so that people can work together more effectively. 3. Represent process in form understandable by humans.	Flowchart (Whatis, 2001) Scenario Analysis (Clemons, 1995) Knowledge-based Techniques (Compatangelo & Rumolo, 1997) Discrete Event Simulation (Pidd, 1988) Qualitative Simulation (Curtis et al., 1992) System Dynamics (Curtis et al., 1992) Swimlane Diagram (Sharp & McDermott, 2001)
Support Process Improvement	1. Identify all the necessary components of a process being modelled. 2. Compare with alternative processes. 3. Estimate the impacts of potential changes to a process before putting them into actual practice.	Petri Nets (Curtis et al., 1992) (Van der Aalst & Heea, 1996) Discrete Event Simulation (Pidd, 1988) System Dynamics (Curtis et al., 1992) Multilevel Modified Petri Nets (Tsalgatidou et al., 1996) Swimlane Diagram (Sharp & McDermott, 2001)
Support Process Management	1. Support development of plans for the project. 2. Monitor, manage, and coordinate the process. 3. Provide a basis for process measurement, such as definition of measurement within the context of a specific process.	Petri Nets (Curtis et al., 1992) (Van der Aalst & Heea, 1996) Plan-based Models (Curtis et al., 1992) Multilevel Modified Petri Nets (Tsalgatidou et al., 1996)
Automated Execution Support	1. Automate portions of the process. 2. Automatically collect measurement data reflecting actual experience with a process. 3. Enforce rules to ensure process integrity.	Functional Models (Curtis et al., 1992) Swimlane Diagram (Sharp & McDermott, 2001)

Table 3.2 BPM Techniques and their Corresponding BPM Goals.

So before choose a specific process modelling technique, we need to identify our modelling goal first, and then select the modelling techniques accordingly. Some of the Business Processes Modelling techniques with their corresponding objectives and goals are summarized in Table 3.2.

3.5 Petri Nets

3.5.1 Introduction

Petri net theory has emerged from the PhD dissertation of Carl Adam Petri entitled "Communication with automata" submitted in 1962 to the faculty of Mathematics and Physics at the Technical University of Darmstadt, Germany.

Although there is an extensive literature on Petri nets, both on the theory and on applications, there is no major introductory textbook available that presents Petri net theory in a consistent manner. Introductory material about Petri nets may be found in Peterson (Peterson, 1981), Reisig. (Reisig, 1992) Additional material can be found in Jensen (Jensen, 1992), Baccelli et al. (Bacceli, Cohen, Olsder, & Quadrat, 1992) Zhou et al (Zhou & DiCesare, 1993).

Petri nets are a modelling and analysis tool that is well suited for the study of Discrete Event Systems. The use of Petri nets leads to a mathematical description of the system structure that can then be investigated analytically.

Since 1980, an International Conference on Application and Theory of Petri Nets has been held every year. Petri nets have been successfully used in the following application areas: Business Process Reengineering, Factory Automation, Performance Evaluation, Communication Protocol Verification, Distributed Software System Specification and Design and VLSI Circuits, etc. Currently, Petri nets are gaining a growing interest among people in Artificial Intelligence because of its adequacy to represent the inference process as a dynamic discrete event system.

Petri net provides a graphical and mathematical modelling tool applicable to many systems (Suzuki, 1990) (Kleyn & Browne, 1993). As a graphic tool, Petri nets can be used as a visual communication aid similar to flow charts. In addition, tokens are used in Petri nets to simulate the dynamic and concurrent activities of the systems.

3.5.2 Fundamentals

A Petri net is a bipartite graph together with an initial state called the initial marking, M_0. Bipartite means that there are two types of nodes. Different symbols are used to distinguish the two types of nodes. By convention, the first type of node is called a place and is denoted by a circle or ellipse. The second type is called a transition and is denoted by a solid bar, or a rectangle. The edges of a Petri net are called arcs and are always directed. The symbols are shown in Figure 3.8. We redraw this figure (Murata, 1989).

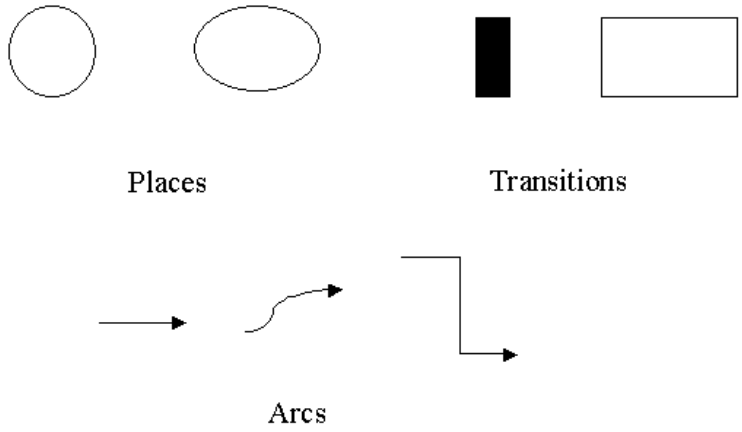


Figure 3.8 Places, Transitions, and Arcs Representation in Petri Nets.

A bipartite graph has a special property: an edge can connect only two nodes that belong to different types. Therefore, there can be an arc from a place to a transition, from a transition to a place, but not from a place to a place or a transition to a transition. Arcs are labelled with their weight, where k-weighted arc can be interpreted as the set of k parallel arcs. Labels for unity weight are usually omitted.

Definition of a Petri net given by Murata (Murata, 1989):

A Petri net is a 5-tuple, PN = (P, T, F, W, M_0) where:

P = $\{p_1, p_2, \ldots p_n\}$ is a finite set of places,

T = $\{t_1, t_2, \ldots t_m\}$ is a finite set of transitions,

$F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs (flow relation),

W: F $\rightarrow$ {1, 2, 3, …} is a weight function,

M_0: P $\rightarrow$ {0, 1, 2, …} is the initial marking,

$P \cap T = f$ and $P \cup T \neq f$.

A Petri net structure N = (P, T, F, W) without any specific initial marking is donated by N. A Petri net with the given initial marking is denoted by (N, M_0).

An example of a Petri Net is shown in Figure 3.9. We adopt this figure from (Peterson, 1978). We denoted it as PN_1. Places are represented by circles and transitions by bars.

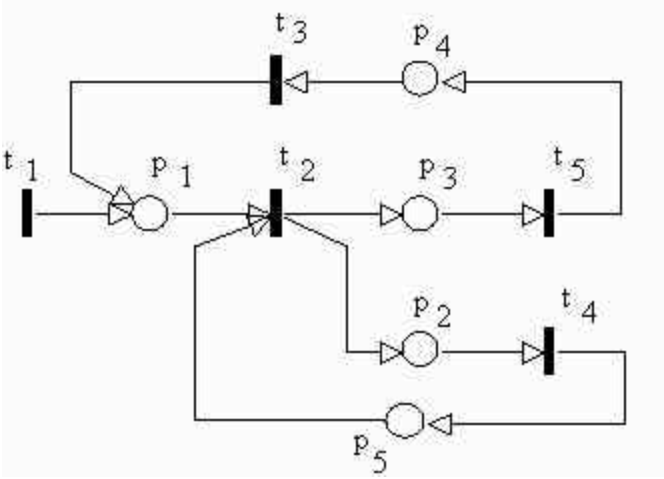


Figure 3.9 Petri Net PN_1.

A marking 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. This is represented by putting k black dots in place *p*. A marking is denoted by M, an m-vector, where m is the total number of places. The *p*th component of M, denoted by M(*p*), is kth number of tokens in place *p*.

In modelling, using the concept of conditions and events, action or activities, places represent conditions, and transitions represent action, events or activities. A transition has a certain number of input and output places representing the pre-conditions and post-conditions of the events or activities, respectively.

The presence of a token in a place is interpreted as holding the truth of the condition associated with the place. If k tokens are put in a place, it indicates that k data items or resources are available. Some typical representations of transitions and their input places and output places are shown in Table 3.3. We adopt this table from (Murata, 1989).

Input Places	**Transition**	**Output Places**
Pre-conditions	Event or Activity	Post-conditions
Input Data	Computation Step	Output Data
Input Signals	Signal Processor	Output Signals
Resources Needed	Task or Job	Resources Released
Conditions	Clause in Logic	Conclusion
Buffers	Processor	Buffers

Table 3.3 Some Typical Interpretations of Transitions and Places.

3.5.3 Firing Rules

According to Murata (Murata, 1989) the behaviour of many systems can be described in terms of system states and their changes. In order to simulate the dynamic behaviour of a system, a state or marking in a Petri net is changed according to the following firing rules:

- A transition t is said to be enabled if each input place *p* of t is marked with at least w(*p*, t) tokens, where w(*p*, t) is the weight of the arc from *p* to t.
- An enabled transition may or may not fire (depending on whether or not the event actually takes place).
- A firing of an enabled transition t removes w(*p*, t) tokens from each input place *p* of t, and adds w(t, *p*) tokens to each output place *p* of t, where w(t, *p*) is the weight of the arc from t to *p*.

A transition without any input place is called a source transition, and one without any output place is called a sink transition. A source transition is unconditionally enabled, and a firing of a sink transition consumes tokens. But sink transition does not produce any tokens.

A pair of a place *p* and a transition t is called a self-loop if *p* is both an input and output place of t. A Petri net is said to be pure if it has no self-loop in it. A Petri net is said to be ordinary if all of its arcs weights are 1's.

These firing rules are illustrated in Figure 3.10, using a simple well-known chemical reaction formula: $2H_2 + O_2 \rightarrow 2H_2O$. Two tokens in each input place in Figure 3.10 (a) show that two units of H_2 and O_2 are available, and the transition of t is enabled at this moment. After firing t, the marking will change to the one shown in Figure 3.10 (b), where the transition t is no longer enabled.

Murata (Murata, 1989) gives an example shown with Petri net PN_2 through Figure 3.11 to Figure 3.14. In Figure 3.11, transition t_1 is enabled, but t_2 is not enabled since one of its input places, p_2, does not contain a token. When t_1 fires, the token is removed from p_1 and a new token is produced in p_2 shown in Figure 3.12. Now, t_2 is enabled, since there is a token in each of its input places, p_2 and p_3. When t_2 fires, the tokens in p_2 and p_3 are removed and new tokens are generated in p_3 and p_4, as shown in Figure 3.13. Note that the marking of the place p_3 does not change after the firing of transition t_2. Transition t_3 is now enabled and it fires. The token is removed from p_4 and a token is generated in p_5 shown in Figure 3.14. No more transitions are enabled and execution of the net is terminated. We redraw Figure 3.10 to Figure 3.14 from (Murata, 1989).

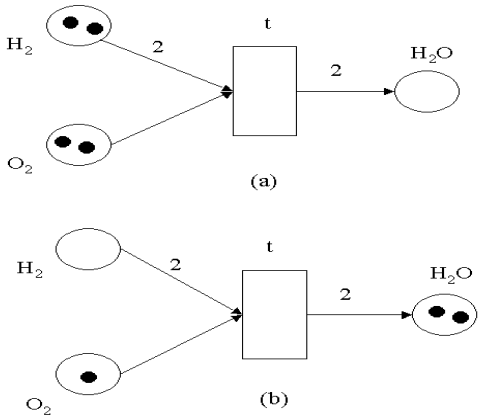


Figure 3.10 An Illustration of a Firing Rule.

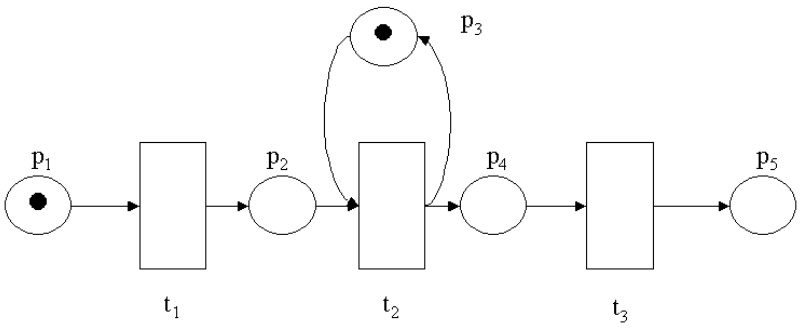


Figure 3.11 PN_2 With Initial Marking.

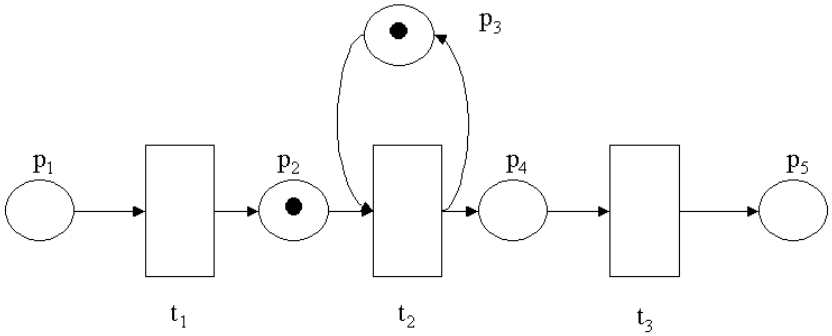


Figure 3.12 The State of PN_2 after One Firing.

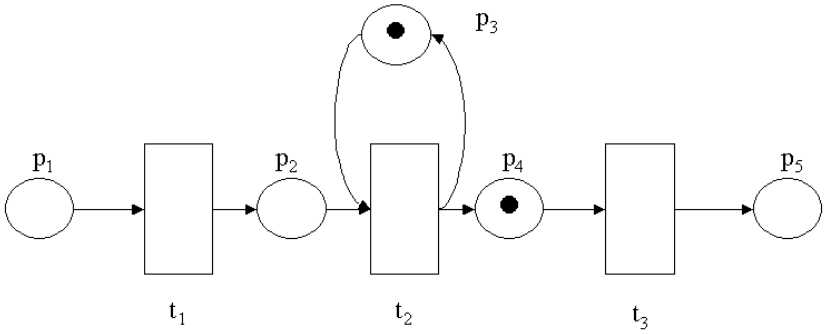


Figure 3.13 The State of PN_2 after Two Firings.

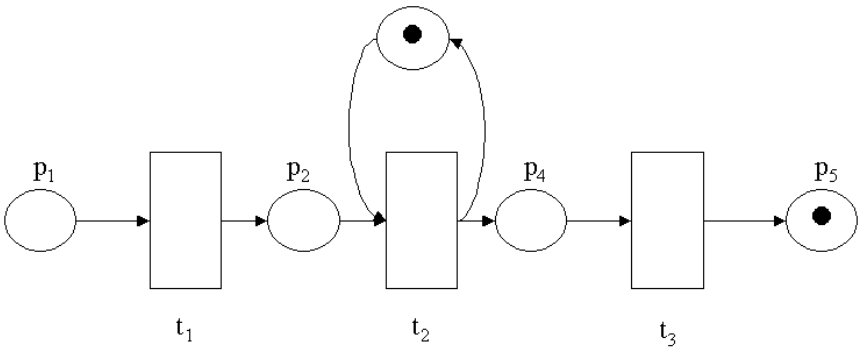


Figure 3.14 The State of PN_2 after Three Firings.

3.5.4 Modelling Example

Communication protocols are one of the areas that Petri nets can be used to represent and specify essential features of a system. The Petri net shown in Figure 3.15 is a very simple model of a communication protocol between two processes.

Process 1 sends a message to Process 2, this message is firstly buffered. At this time, Process 1 waits for the Ack from Process 2. Transition “Receive Message” fires after

Process 2 receives the message from Process 1 then it sends an Ack back to Process 1. When Process 1 receives the Ack, one round of communication finished. System is ready for the next round of communication.

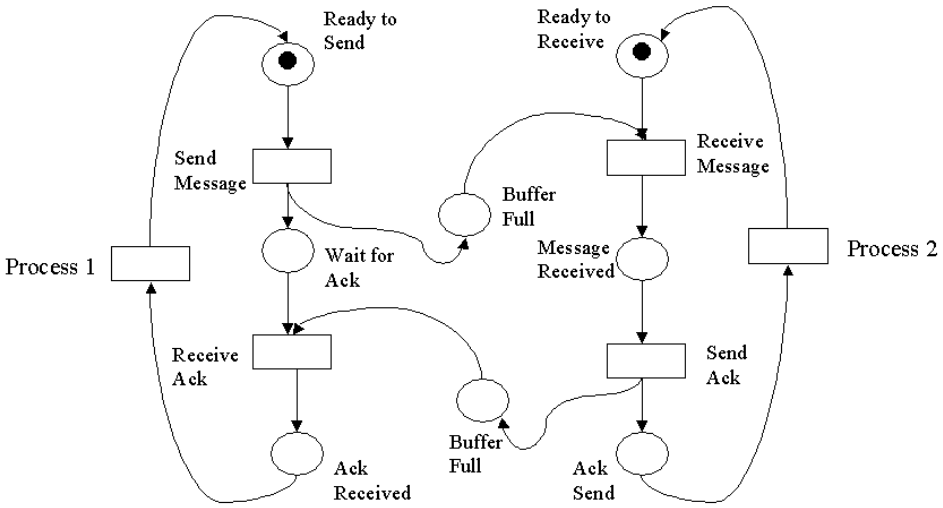


Figure 3.15 A Simplified Petri Net Model for Communication Protocol.

3.5.5 Summary

Petri net is an abstract, formal model of workflows. The properties, concepts, and techniques of Petri nets are being developed in a search for natural, simple, and powerful methods for describing and analysing the workflows and control in real world systems.

The Petri net graph models the static properties of a system, much as a flowchart represents the static properties of a computer program. In addition to the static properties represented by the graph, a Petri net has dynamic properties that result from its execution (Peterson, 1978). Since a finite state machine can be modelled by a Petri net, a regular language is a Petri net language. It has been shown that all Petri nets languages are context-sensitive languages (Murata, 1992).

Petri nets have been proposed for a very wide variety of applications. This is due to the generality and permissiveness inherent in Petri nets. They can be applied informally to any area or system that can be described graphically like flow charts and that need some means of representing parallel or concurrent activities. However, careful attention must be paid to a trade-off between modelling generality and analysis capability according to

(Murata, 1992). That is, the more general the model, the less amenable it is to analysis. In applying Petri nets, it is often necessary to add special modifications suited to the particular application.

3.6 Technology for Business Process Reengineering

For Business Process Reengineering, new technologies for the execution of business process are needed to support system dynamics, customisation, and integration. Workflow Management promised to provide a suitable infrastructure for the business process in a distributed environment. Workflow Management System aims at the automation of business process.

3.6.1 Workflow Management System

Workflow Management Systems (WFMS) (Allen, 2001) are used to automate business processes. They control the flow of work through a company, thus providing the right person with the right task at the right point of time. This helps in streamlining the business processes of a company and has the potential to improve the productivity of recurring tasks by a significant amount.

Workflow Management Systems results in a gain in efficiency, productivity, enhanced reactivity, clear progress reports as well as quality and cost control benefits. Table 3.4 summarizes some of the benefits we can get from Workflow Management System.

3.6.2 Workflow Management Coalition

The WFMC (WFMC, 2001) is an international organization whose mission is to promote workflow and establish standards for Workflow Management Systems. The WFMC was founded in 1993. In January 1995, the WFMC released a glossary, which provides a common set of terms for workflow vendors, end-users, developers, and researchers.

In this glossary, workflow is defined as:

> The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules.

Roles	Benefits From Workflow Management
Company	Cost and Performance Measurement Quality control Confidentiality and Access Control Adherence to Procedures
Agent	Clear Picture of Tasks Automatic Access to tools Information on Context of Tasks
Client	Response Time Services Quality Agent Accessibility Information on Case Status
Manager	Just in Time Display of Tasks Just Enough Information Warning System Measurement and Tracking of Quality

Table 3.4 Benefits from Workflow Management.

Workflow Management System is defined as:

> A workflow management system defines, creates and manages the execution of workflows, through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants, and, where required, invoke the use of information technology (IT) tools and applications.

A workflow engine provides the run-time execution of business processes. Engines can be embedded within other applications or they can be deployed as independent applications inter-operating with other applications.

From these definitions we can see that the main purpose of a Workflow Management System is to support of the definition, execution, registration and control of processes. The ultimate goal of Workflow Management System is to make sure that the activities are executed by the intended role in the right order.

3.6.3 Type of Workflows

Four categories of workflows are identified by Alonso et al (Alonso, Agrawal, Abbadi, & Mohan, 1997) and Van der Aalast (Van der Aalst, 1998). The parameters used for

this classification are the similarities among the business processes involved and their value to the associated enterprises. These four categories are: administrative, ad hoc, collaborative, and production workflows.

Administrative workflows refer to bureaucratic processes where the steps to follow are well established and there is a set of rules known by the participants. Course Registration in a University is a typical administrative workflow. The most important feature of an administrative workflow system is the ease to define the process.

Ad hoc workflow systems allow users to create and amend process definitions very quickly and easily to meet circumstances as they arise. It is quite similar to administrative workflow except that ad hoc workflows are created to deal with exceptions and unique situations. From a university's point of view, the process of applying for a degree is an administrative procedure, for a student it is something that happens only once and is therefore ad hoc.

Collaborative workflow focuses on teams working together towards common goals. Groups can vary from small, project-oriented teams, to widely dispersed people with interests in common. A collaborative workflow may involve several iterations over the same step until some form of agreement has been reached, or it may even involved going back to an earlier stage. A good example is writing a paper by several authors.

Production workflow can be characterized as the implementation of critical business processes. The key goal of production workflow is to manage large numbers of similar tasks, and to optimise productivity. This is achieved by automating as many activities as practical. Human input is required only to manage exceptions in case those work items fall outside pre-determined process tolerances.

Production Workflow can manage hugely complex processes, and can be tightly integrated with existing systems. Production workflow is optimised to attain high levels of quality and accuracy by executing highly repetitious tasks, usually in a non-stop manner. Credit and loan applications are a typical production workflow process.

Alonso et al (Alonso et al., 1997) also discuss the difference between administrative and production workflows. The authors emphasize that the main points to consider are the large scale, the complexity and heterogeneity of the environment, the variety of

people and organizations that involved, and the natural of the tasks. Production workflows tend to be executed over heterogeneous systems, frequently legacy applications, and monitoring tools should be provided to do statistical analysis the execution of these processes.

In our thesis, we are interested in production workflows, because the workflows we are going to study in international trade belong to this category. Figure 3.16 and Figure 3.17 illustrate the classification of workflow management systems and the relationship between them through different perspectives. We adopt Figure 3.16 and Figure 3.17 from (Alonso et al., 1997).

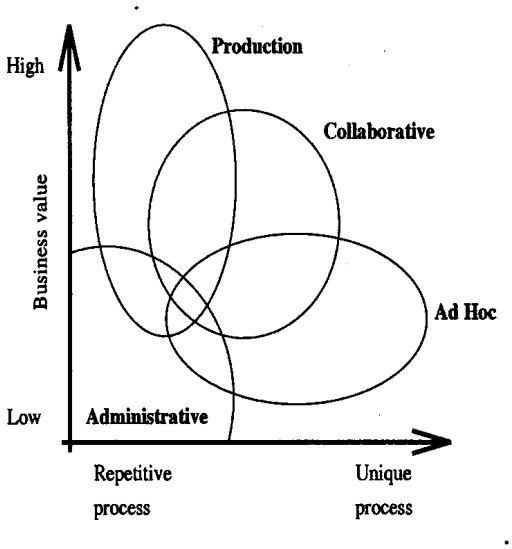


Figure 3.16 Classification of Workflow Management Systems.

3.6.4 Jablonski's Methodology

Jablonski (Jablonski, 1995) discusses the interrelationship between Business Process Modelling and Workflow Management and presents a methodology for integrating business process models and workflows.

The main benefit of this approach by Jablonski is to maintain a direct relationship between business processes and workflows. This helps to directly pass on modifications of a real world business process to the corresponding workflow. This feature is very important to sustain consistency between business processes and workflows. So we can instantaneously react to changes in the business processes and could adjust the corresponding workflows.

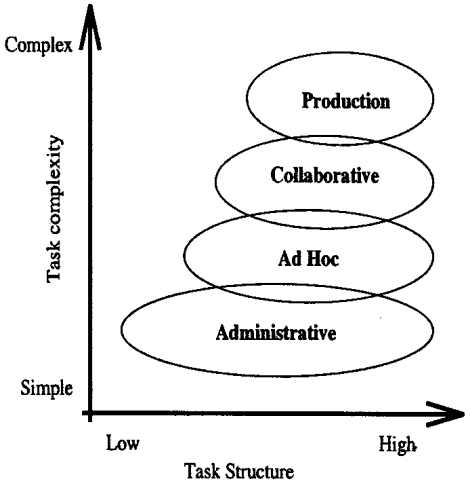


Figure 3.17 Classifications of Workflow Management Systems.

Another advantage of this approach is: matured capabilities of Business Process Modelling tools and the powerful execution capabilities of Workflow Management System can be used in an integrated manner. This mutual benefit exactly fulfils the requirement of modern Business Process Reengineering efforts: to maintain a very tight connection between real world modelling and model execution in order to react fast to changes in the business environment.

The methodology is based on the notion of meta-models that are used to provide a generic mechanism for translating between business models and workflow specification. The mapping of business process to workflow would be a necessary step towards holistic approach to business process management in organizations. Here the meta modelling by Jablonski (Jablonski, 1995) is defined as:

> The process of specifying the requirements to be met by the modelling process or establishing the specifications, which the modelling process must fulfil.

Workflow Management technology is claimed to be the one of the innovative applications in 1990s (Schal, 1996). It is a broad term, used in a number of different contexts and environments. It has the ability to reflect the business process rather than to support or automate just discrete tasks. This will result in an improved productivity and flexibility needed for business competitiveness.

The need for reorganization and improvement of business process and the advances in information technology creates a huge market request for workflow systems. Modern Workflow Management Systems enable the design, enactment, and monitoring of workflows in different sorts of environment, such as heterogeneous and distributed, allowing efficient process execution and management. (Casati, Ceri, Pernici, & Pozzi, 1998)

3.7 Summary

During the last several decades, the market structure has changed tremendously. The market is driven by customers instead of suppliers. Classical business structures are no longer suitable in a world where competition, customers and change demand flexibility and quick response. In order to survive and prosper in today's global economy, companies are being forced to re-evaluate the way they do business with their customers and their vendors. It is necessary to deliver high quality products and services to gain and maintain customer loyalty and reduce costs. This makes Business Process Improvement a very hot topic.

To improve a business process, we have three models: Continuous Process Improvement, Business Process Reengineering and their combination Process Management. Successful Business Process Reengineering work on business process may bring about rapid change and dramatic improvement to the current business process, so we focus on this model in this thesis. But in many real-life business change projects, we saw a high failure rate.

Some of the researchers pointed out that the shortage of a comprehensive, scientifically grounded design methodology to guide Business Process Reengineering efforts, difficulty in capturing existing processes in a structured way, and inability to recognize the dynamic nature of the processes are the main reasons result in high failure rate in Business Process Reengineering projects.

In this chapter, we introduced some techniques currently being used in Business Process Reengineering projects such as Flowchart, Scenario Analysis, System Dynamics, Knowledge-based Techniques, Message Sequence Chart, Petri net and Swimlane Diagram. Each technique has its characteristics and is suitable for a special kind of application.

Compared with other techniques, Petri net has formal semantics and powerful analysis methods; it is easy to understand, and can represent business process in a precise and unambiguous way. The graphical representation of Petri nets is suitable for visualizing business process in a comprehensive way, and supporting the communication among the Business Process Reengineers.

Workflow Management technology can be used in a number of different contexts and environments. It has the ability to reflect the whole business process rather than to support just discrete tasks. This results in an improved productivity and flexibility needed for Business Process Reengineering.

Matured capabilities of Business Process Modelling techniques and the powerful execution capabilities of workflow Management System can be used in an integrated manner. These mutual benefits have the potential to maintain a very tight connection between real world modelling and model execution in order to react fast to changes in the business environment.

In the next two chapters, we will first examine the previous work done by Professor Lee on international trade. We then set up a simple business process model and use the techniques and technologies we introduced in this chapter to model and execute our business process model.

Chapter 4 Reengineering of International Trade

In the previous chapter, we discussed why business processes need to be improved. Also we reviewed some techniques and technology in Business Process Reengineering. In this chapter we discuss how Business Process Reengineering can be applied to international trade as an example to show how a business process can be reengineered.

In our survey of the literature on reengineering of international trade, we found just one active group. This group is headed by Professor Lee of Erasmus University, in Rotterdam, the Netherlands.

Professor Lee (Lee, 1999) identifies a common problem in international trade: each party may find it difficult to know whether the other party is trustworthy or not as they have no prior trading relationship. These trading parties who don't know each other, may come from different countries and cultures. In order to facilitate the international trade, trading parties' interests should be protected in the event of accident, negligence, or international fraud. We call this set of issues the trustworthiness problem.

In order to assure the interests of the trading parties involved in international trade procedure, Professor Lee introduced a new concept called an electronic trade scenario. Trade scenarios govern the activities of all the parties involved in a set of related business transactions. It is believed that electronic trade scenario is a potential solution to the trustworthiness problem we mentioned above. An electronic trade scenario is specified by a graphical representation language called Documentary Petri Nets. A case-tool called InterProcs is also made available by Professor Lee to assist in designing and prototyping trade scenarios.

Unfortunately, Professor Lee did not discuss much in detail on how a business process can be mapped using the Business Process Modelling techniques we introduced in Chapter 3. So in the rest of this chapter, we set up a very simple trade scenario called GT3 by simplifying the Documentary Credit Procedure we discussed in Chapter 2. We

named “GT3” our scenario because it is for Global Trading, or international trade, with three roles defined in the scenario. Three roles are: Importer, Exporter, and Transporter. An Importer buys products or services from an Exporter; a Transporter is responsible for the delivering of products.

For better understanding of our GT3 model, we use several Business Process Reengineering techniques to analyse it from a different point of view. Also, we modify Professor Lee’s Documentary Petri Nets by subtracting some constraints and obtain a simplified representation language called Linear Documentary Petri Net. Then we use Linear Documentary Petri Net to represent our GT3 model for further study.

4.1 Review of Work Done by Professor Lee

Professor Lee (Lee, 1999) introduced the concept of electronic trade scenario as a potential solution to the trustworthiness problem we mentioned above in the open electronic commerce environment. Parties intending to conduct electronic commerce have to know about one another’s way of doing business before they can exchange electronic data.

According to Professor Lee, a trade scenario is the mutually agreed-upon set of procedures, documents and rules that governs the activities of all the parties involved in a set of related business transactions. This trade scenario controls all the interactions among the trading partners. The scenarios specify which action is to be performed by which party, and the order in which these actions are to be undertaken.

In Professor Lee’s point of view, the contracting process in a trading procedure is divided into three main phases: shopping, negotiation, and performance as illustrated in Figure 4.1. We redraw this figure from (Lee, 1999). The shopping phase involves navigating among the products offered by various vendors, after that the customer identifies a prospective vendor and specifies the product characteristics to be purchased.

In negotiation phase, the trading parties need to exchange required documents, transfer payment, and deliver products between them. The output of this negotiation phase will be the specific trade scenario that involving parties will undertake for this particular contract. The documents passing among the parties will be standardized EDI documents. In some contract domains, additional controls are needed, for example,

secured loan, such as home mortgages. Additional control documents such as pertaining to the borrower's credit-worthiness and inspection of the asset are needed.

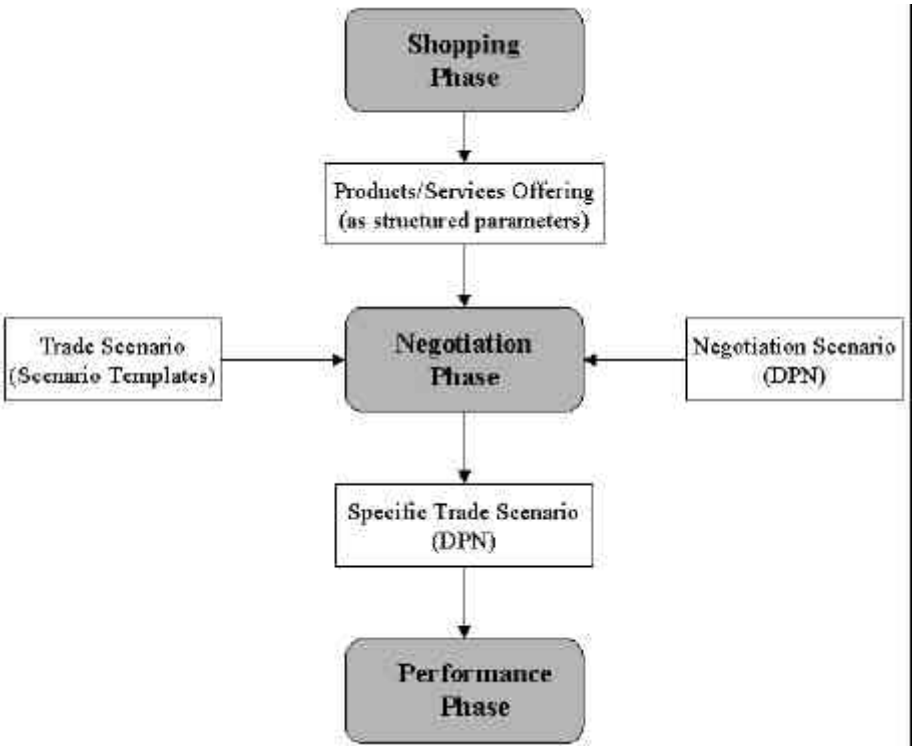


Figure 4.1 Contracting Procedures from Professor Lee's Point of View.

In this section we introduced the common problem identified by Professor Lee (Lee, 1999) in international trade: lack of trust among trading parties because they have no prior trading relationship in open electronic commerce environment. Professor Lee invents electronic trade scenario as a potential solution to this problem. These trade scenarios are mutually agreed-upon procedures, documents and rules that govern the activities of the trading partners. Trade scenarios can be downloaded by trading partners as needed from a publicly accessible global repository for a particular trade procedure so as to assure the interests of the trading partners involved in this particular transaction.

In order to represent this electronic trade scenario, Professor Lee advocated Documentary Petri Nets (DPN) to specify the procedural aspects of a trade scenario. The use of Documentary Petri Nets will be studied in the next section.

4.1.1 Professor Lee's Techniques: Documentary Petri Nets

According to Professor Lee (Lee, 1999), one basic issue is how the trade scenario should be represented. In Professor Lee's point of view, there should be three kinds of preliminary requirements on the representation of electronic trade scenario. They are formal requirements, notational requirements, and verification requirements. In the following several paragraphs, we will discuss these three requirements in detail.

Formal requirements: according to Bons et al (Bons, Lee, Wagenaar, & Wrigley, 1995), formal requirements on representation of a trade scenario include the possibility to express concurrency, choice and contingency, and deontic relationships.

- Concurrency: In a trading procedure, the multiple involved parties may perform their actions in a parallel way, so the ability of dealing with concurrency is essential for this representation language.

- Decision points: The execution of a trade scenario often depends on some internal or external decisions, so the modelling constructs for such decision points should be included in the representation language.

- Deontics: According to (Hilpinen, 1971), deontic logic is a branch of logic that formalizes reasoning about normative vs. non-normative behaviours by means of primitives such as obligations, prohibitions and permissions. Bons et al (Bons et al., 1995) state that the representation of deontic in a trade scenario is essential because involved parties should be able to derive their obligations, rights, and duties at each point during the execution of the trade scenario. For example, sending a purchase order to a seller will in most cases bring about an obligation for the buyers to buy products from the seller.

- Time: Bons et al (Bons et al., 1995) think that the modelling of absolute time is necessary in a trade scenario, because the parties involved in the trading procedures may want to indicate an absolute deadline for a specific trade. For example in just-in-time logistics, time is one of the most critical factors to be investigated.

Notational requirements include the possibility to represent trade scenario design in a graphical way. And also a hierarchical way is essential to decompose the trade scenario into a number of levels.

- A graphic representation: as Streng (Streng, 1993) points out, several intangible properties of EDI linkages cannot be validated using quantitative techniques. He argued that some properties can only be evaluated using graphical/animated dynamic modelling techniques such as management information, project uncertainty, etc. A graphical representation of trade scenario is essential because it enables business specialists to reason about the proposed trade scenarios without the need to become experts in formal representation languages.

- A hierarchical (de)composition: This hierarchical decomposition of trade scenario is necessary for us to reduce the complexity of the specifications. Another advantage is that some parts of the specification can be reused by other trade scenarios.

Verification requirements include the possibility of automated verification according to Bons et al (Bons et al., 1995). Techniques that can be used to perform verifications include formal algorithms, pattern recognition, gaming theory, etc. Formal algorithms can be used to prove the absence or presence of deadlock states, never-ending loops. Pattern recognition can be used to reason about control issues. Gaming can be used to verify dynamic properties.

With these requirements in mind, Professor Lee and his research team examine many possible representations, included state-transition diagrams, marked graphs, event nets, event grammars, the event calculus, process algebras, and temporal and dynamic logics. They eventually found Petri nets to be an acceptable candidate that captures the temporal/dynamic aspects of electronic trade procedures (Lee, 1999). It offers both a graphic representation and a formal basis for verification of various computational properties.

Professor Lee (Lee, 1999) and Van der Alast (Van der Aalst, 1998) both state that Petri nets are suitable for representation purpose in a great variety of problem domains, such as workflow systems, where sequence, contingency, and concurrency of activities need to be modelled.

Van der Alast (Van der Aalst, 1992) states that Petri nets are especially suitable for modelling discrete dynamic systems. Because it offers various kinds of both formal and informal analysis methods in addition to its capability to graphically model both concurrency and choice.

A classic Petri net as we introduced in the previous chapter, is a bi-partite, directed graph. It has two disjunctive sets of nodes: places and transitions. Arcs connect the places and transitions or vice versa. The dynamic behaviour of the modelled system is represented by tokens flowing through the nets. A transition is enabled if all its input places contain at least one token. If so, the transition removes one token from each input place and instantaneously produces one in each output places. This is called the firing of the transition.

However, classic Petri nets by themselves offer only a temporal framework for procedural representation. So Professor Lee (Lee, 1999) thinks it is necessary to add extensions to the classic Petri nets representation to make it more appropriate for the modelling of trade scenario. The extended Petri net is termed as Documentary Petri Nets (DPN).

Documentary Petri Nets (DPN) interpret transitions in usual Petri nets as the actions of contracting parties in international trade which are indicated by an associated label of the form shown in Figure 4.2. We redraw this figure from (Lee, 1999). In usual Petri nets, system state changes via the firing of a transition; in Documentary Petri Nets, when a role performs an action, documents or goods or payment will be sent out to other roles, the system is getting closer to the ending of this particular trading procedure.

Classic Petri nets model time only in relative terms. In Documentary Petri Nets, absolute time notations for deadline are added. A special kind of transition called a timer event, with the associated label shown in Figure 4.3. The “>>” operator is used to compare date/time terms. Usually, the built-in parameter, @current_date, is compared with another date/time terms such as a delivery_deadline. A sample structure is shown in Figure 4.4.

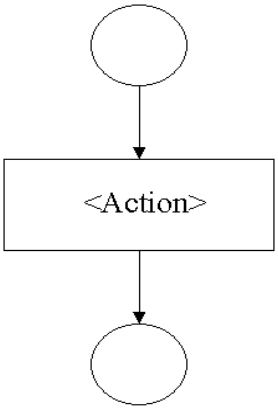


Figure 4.2 DPN Transition Syntax (Lee, 1999).

In Figure 4.4, the black token will be taken from the input place at the top of the figure by the first transition to occur. Thus, if the Delivery_Goods occurs after the delivery_deadline, then the state of breach will be reached.

An important extension added to classic Petri net is the representation of documents. Professor Lee (Lee, 1999) adds another kind of place node, called a document place, drawn as a rectangle. In each role procedure, each document place has an associated label to identify the name, sender, and receiver of the document. This is shown in Figure 4.5.

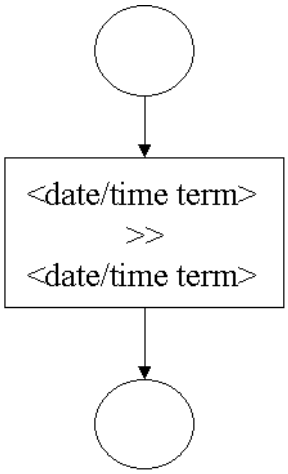


Figure 4.3 DPN Timer Event Syntax (Lee, 1999).

Usually the document list will be only a single document type, but Professor Lee's syntax also allows multiple documents to be sent in a single documentary communication.

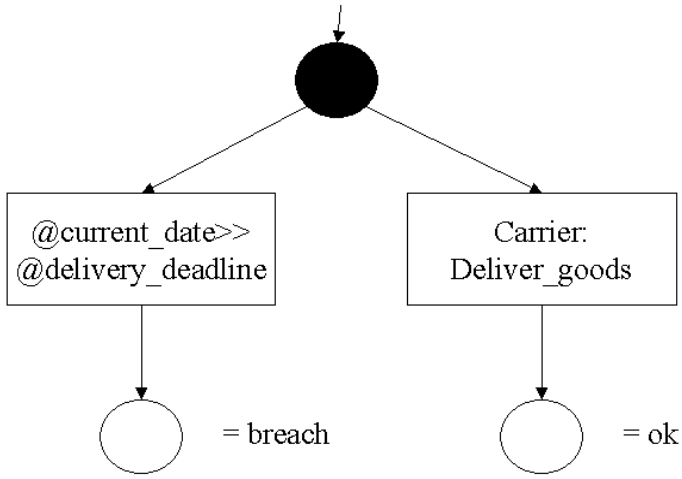


Figure 4.4 Example DPN for Deadline (Lee, 1999).

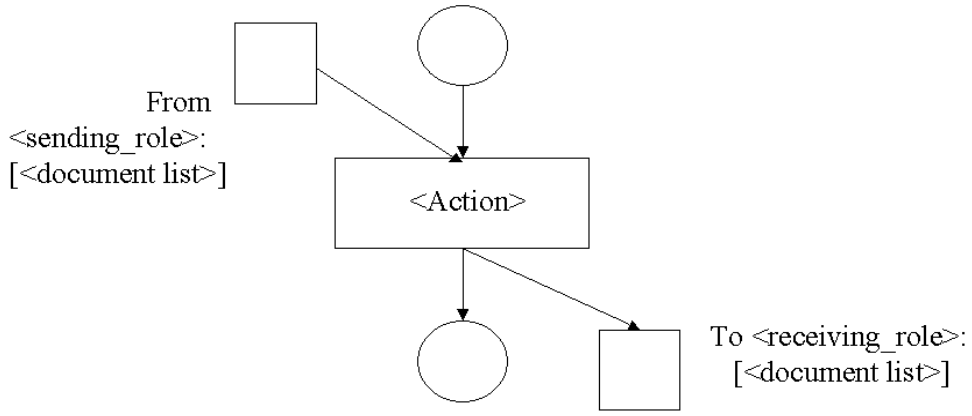


Figure 4.5 DPN Document Place Syntax (Lee, 1999).

Professor Lee (Lee, 1999) indicates that in the trade scenario, a frequent type of documentary exchange is the transfer of funds. This special kind of document will be sent to a bank, such as a payment instruction. For international trade, although it is fairly rare to exchange cash in business-to-business transactions, the same notation of a "document" places is applied to the cash payment. The amount and currency of the fund transfer is specified in the structure of these documents.

Another variation of document places that is occasionally used is a goods place, indicating the exchange of physical goods. The notation for the physical goods is a cube, and it's labelling is similar to that for a document place, as shown in Figure 4.6.

In addition to the above-described labels, any of the control places, document places, or goods places may have an additional kind of label known as a state predicate. Professor Lee (Lee, 1999) points out that the same predicate notation as in Prolog is used in these additional labels to indicate additional properties and relations that become true when the place is marked. They are commonly used to indicate changes in deontic status, for instance, obligation (X: A, Date), which means that party X has an obligation to perform action A before the deadline Date. This is illustrated in Figure 4.7.

Until so far, we can see that in Documentary Petri Nets, a state transition is enabled by receiving a document, goods, or funds, or the expiration of an internal timer. Firing a transition can lead to sending documents, goods, funds, or setting an internal timer.

Professor Lee indicates that one important aspect of modelling trade scenario by Documentary Petri Nets is the ability to model the procedure of each role as a separate DPN. This allows the decomposition of a trade procedure into a number of logically separate sub-nets. This modelling style results in a clear, visual separation of the various roles that also enables their geographical separation.

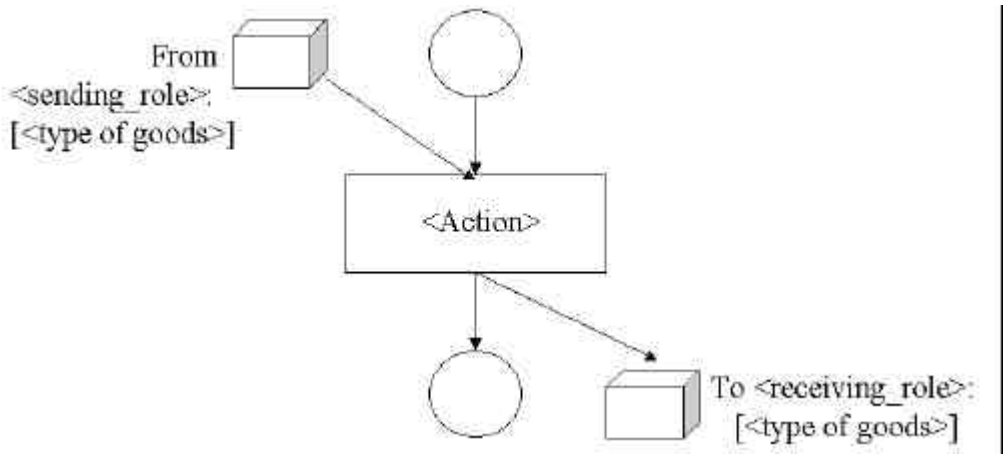


Figure 4.6 DPN Physical Goods Syntax (Lee, 1999).

Professor Lee emphasizes that it is just this separation characteristic of the Documentary Petri Nets that allows legally autonomous parties to execute automated trade scenario in a distributed fashion. The only coordination between the various role scenarios is by the passing of electronic documents they exchange.

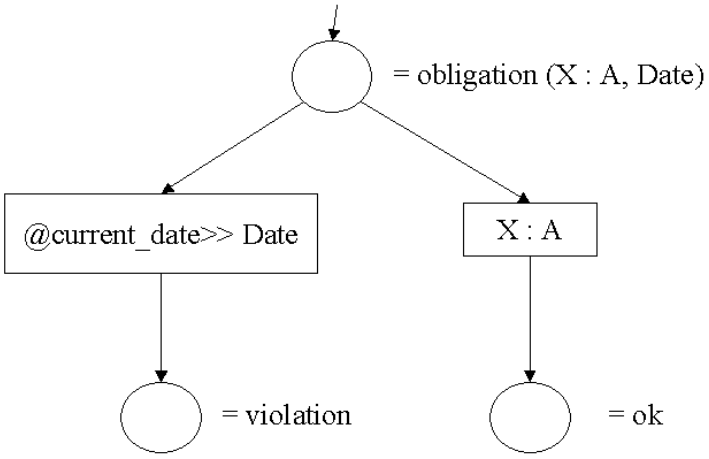


Figure 4.7 DPN Example: Deontic Status Labels on Control Places (Lee, 1999).

In this section we have a close look at Professor Lee's Documentary Petri Nets. Just as Professor Lee indicates (Lee, 1999) that this Documentary Petri Nets are quite suitable to be used as a representation language for modelling electronic trade scenario.

In order to validate the concepts and representation for electronic trade scenario, Professor Lee et al (Lee, 1999) developed InterProcs to demonstrate the feasibility of Documentary Petri Nets through realistic prototype transaction models with actual EDI documents. In the next section we will discuss the characteristics of InterProcs.

4.1.2 Professor Lee's Technology: InterProcs

According to Professor Lee (Lee, 1998), the motivation behind the development of InterProcs is to validate the concepts and representations for electronic trade procedures, demonstrating their feasibility through realistic prototype transaction models with actual EDI documents.

InterProcs system divides into two separate systems: InterProcs Executor and InterPorcs Designer. InterProcs Executor is used to execute an existing trade scenario, while InterProcs Designer is used to design trade scenarios.

- InterProcs Designer

The general design methodology of a trade scenario given by Professor Lee is illustrated in Figure 4.8. Design a trade scenario, we start executing InterProcs in Designer mode, and then follow the steps Professor Lee recommended. UseCase Diagram is used to define roles involved in the particular scenario. Unified Modelling

Language (UML) is a standard notation for the modelling of real-world objects as a first step in developing an object-oriented design methodology. Sequence Diagram is used to define the order of the documents among roles.

Activities Diagram is used to define the actions each role need to perform in the scenario. Joint Procedure Diagram is to get an overview of the resulting trade scenario with all roles in it. XML Schema is used to design the content of the documents. Extensible Mark-up Language (XML) is a flexible way to create common information formats and share both the format and the data on the web. For the present, Professor Lee uses the Schema for Object-oriented XML (SOX) approach for documents. Role Procedure is used to illustrate each role to be represented as a separate Documentary Petri Net.

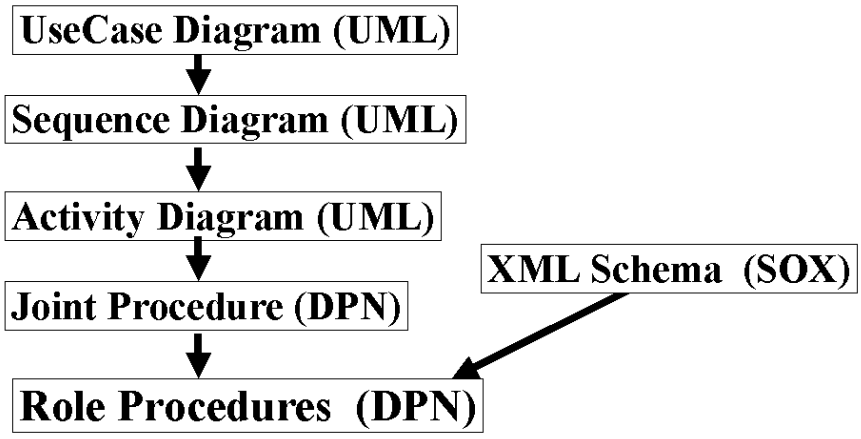


Figure 4.8 Lee's Methodology for Trade Scenario Design (Lee, 1999).

Figure 4.9 to Figure 4.14 illustrate the steps we go through while designing a simple trade scenario by top-down approach as described in Figure 4.8. In this trade scenario we have two roles, buyer and seller. Buyer order goods from a seller, first buyer send a request for quote to seller; when seller receive the request, it will send a quote to the buyer after processing the request; then buyer receive the quote and make its final decision on purchase. The resulting automated trade scenario is shown in Figure 4.14.

In Use Case Diagram designing phase shown in Figure 4.9, we define that there are two roles involved in this trade scenario called order_goods. The two roles are buyer and seller. In Sequence Diagram shown 4.10, we define the names and order of the documents between these two roles. RFQ is first sent to seller, and then at a later time

Quote is sent back to Buyer. We can easily identify this Sequence Diagram is the same as the Message Sequence Chart (Mauw & Reniers, 1994) we introduced in Chapter 3. They both describe the communication behaviours among parties and focus on the order of the message.

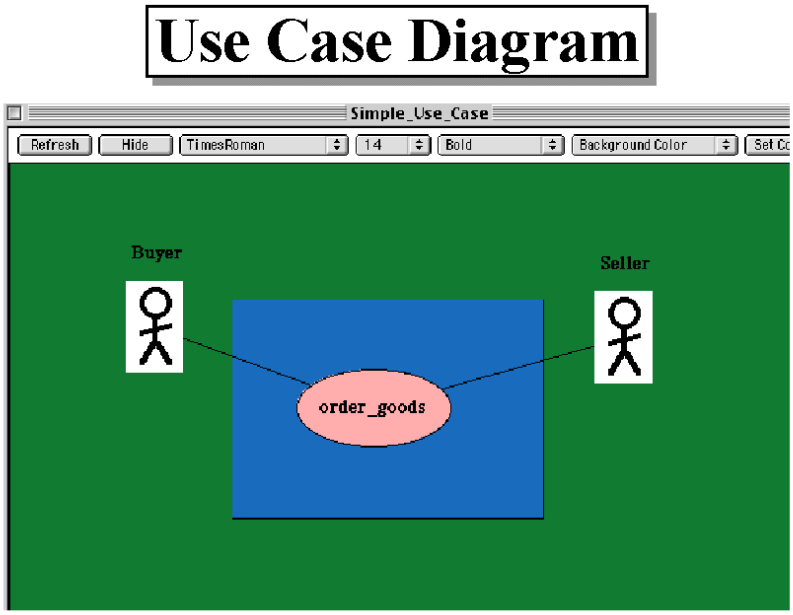


Figure 4.9 Use Case Diagram Design Using InterProcs.

In Activity Diagram shown in 4.11, we define the actions each role need to perform in this trade scenario. For example, Buyer needs to perform two actions in this case. One is send_request_quote, the other is receive_quotation. This Activity Diagram is quite similar to Swimlane Diagram we introduced in Chapter 3. They both focus on the actions or activities of each role and the relations between these actions or activities. They both have the ability to depict the system at any level of abstraction. At this stage, a Joint Procedure diagram shown in Figure 4.12 can be generated by InterProcs Designer.

XML Schema is mainly used to describe the content of a document. Figure 4.13 illustrate all the contents should be included in the purchase_order for this specific trade scenario. To generate Role Procedures from Joint Procedure automatically is a built-in function in InterProcs Designer.

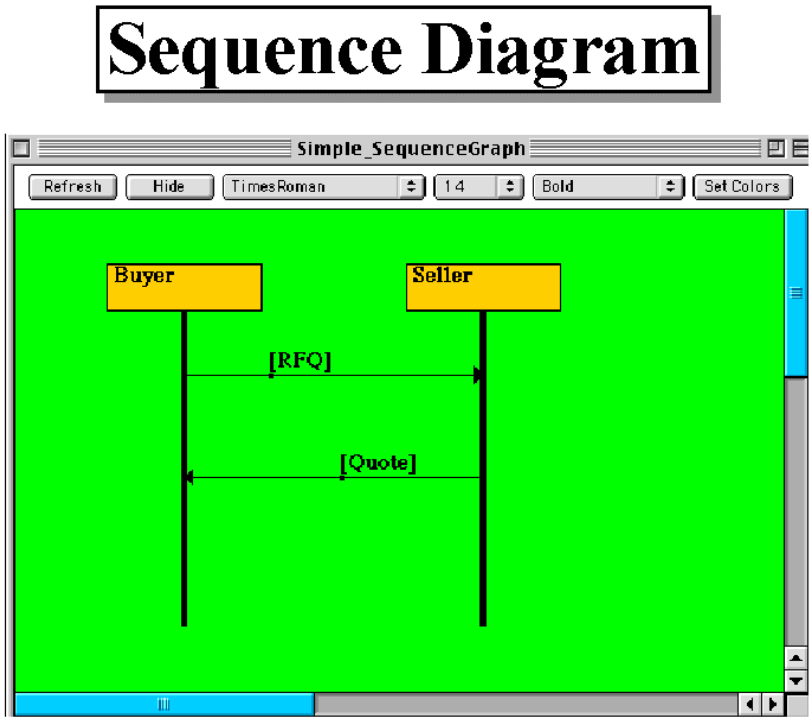


Figure 4.10 Sequence Diagram Design Using InterProcs.

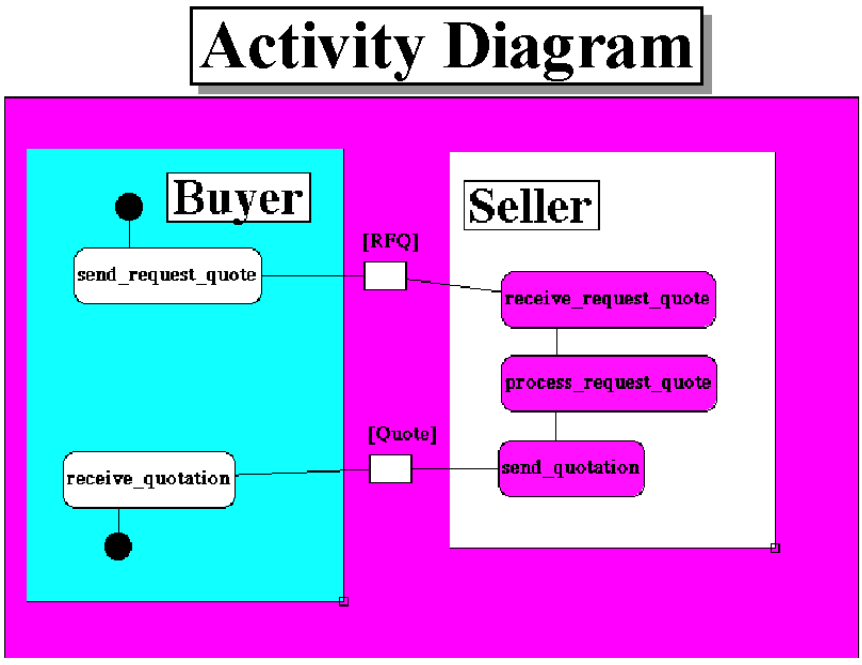


Figure 4.11 Activity Diagram Design Using InterProcs.

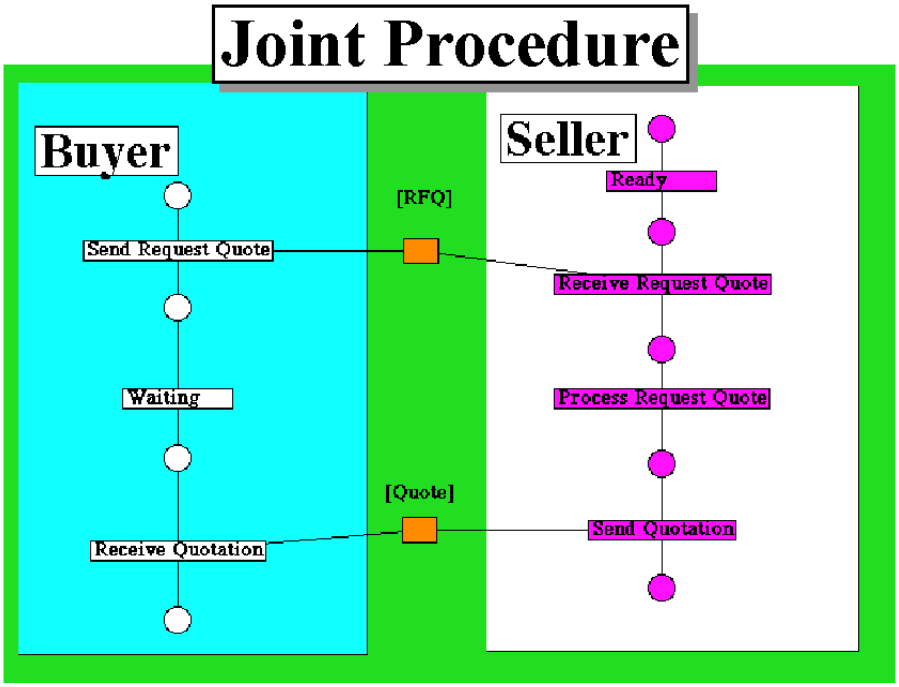


Figure 4.12 Joint Procedure Diagram Design Using InterProcs.

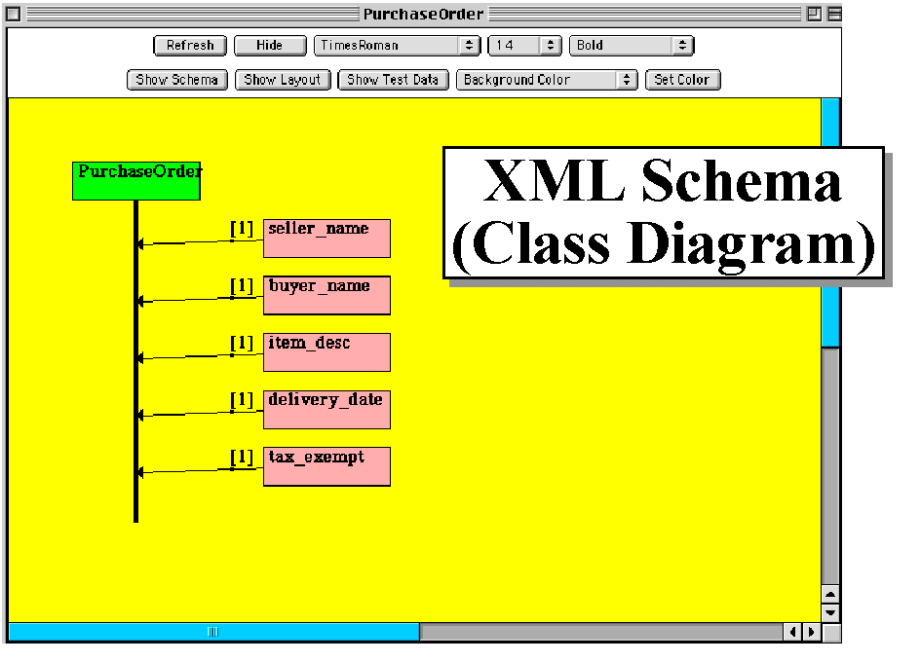


Figure 4.13 XML Schema Diagram Design Using InterProcs.

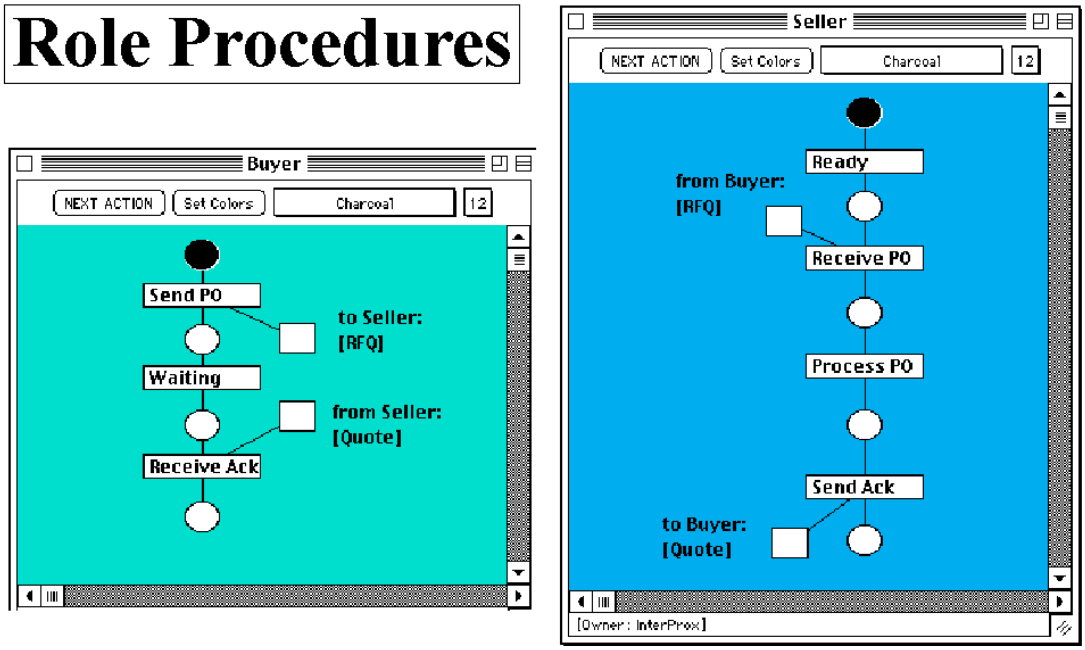


Figure 4.14 Role Procedure Diagram Design Using InterProcs.

- InterProcs Executer

In InterProcs Executor, a trade scenario can be viewed and executed in three modes. They are Viewer Mode, Gaming Mode, and Network Mode.

Viewer mode provides a simulation of the total scenario on a single machine. All roles in this scenario are displayed together in a single screen. Documents are transmitted among the roles internally within the executing program. This viewer mode is very useful for design and demonstration purpose. The designers can put a new version of the scenario as an applet on the web to show their client on a remote location and get feedback easily.

In Gaming mode, the roles of a trade scenario can be downloaded and executed by multiple machines in a LAN. This mode is intended for interactive gaming and testing. For instance, in a group collaboration setting, manager can play different roles and evaluate the scenario from the different role perspectives.

The purpose of Network mode is for prototype testing of the trade scenario in geographically separate locations. Roles of the trade scenario can be downloaded and executed using InterProcs Executor as a stand-alone application. Only one role can be

viewed to any single user. Documents are transmitted among the roles via normal E-mail. Different parties can participate in the testing of the scenario from their local sites.

Professor Lee gives an example of a simple trade scenario being executed in InterProcs Executor. The screenshot of joint_activity_graph of this simple scenario is given in Figure 4.15. A screenshot of role procedures for this simple scenario is illustrated in Figure 4.16. We adopt these two figures from (Lee, 1999). In this simple scenario, there are two roles involved, buyer and seller. And two documents between the two roles. One is purchase order, another is a corresponding acknowledgement.

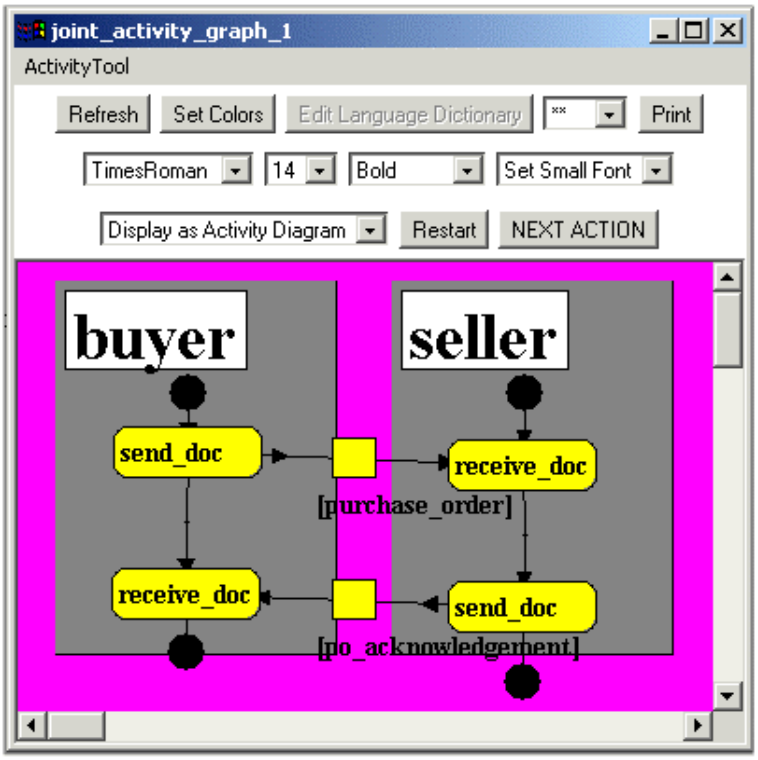


Figure 4.15 Screenshot of Joint_Activity_Graph in InterProcs Executer.

In international trade, documentary credit procedures usually involve numerous agents who often must interact in different languages, as we mentioned before. InterProcs can provide multiple versions of documents in various languages. Figure 4.17 illustrates a sample snapshot of EDI document in Dutch.

The embedding of the concept of audit daemons into InterProcs is in progress by Lee and his research team. Audit daemon was originally developed by Professor Lee in 1990 (Lee, 1992). Basically, the purpose of an audit daemon is to inspect a procedure

and identify possible control weakness and potentials for fraud. Audit daemons consist of deductive rules, and special pattern-matching predicates for Documentary Petri Nets.

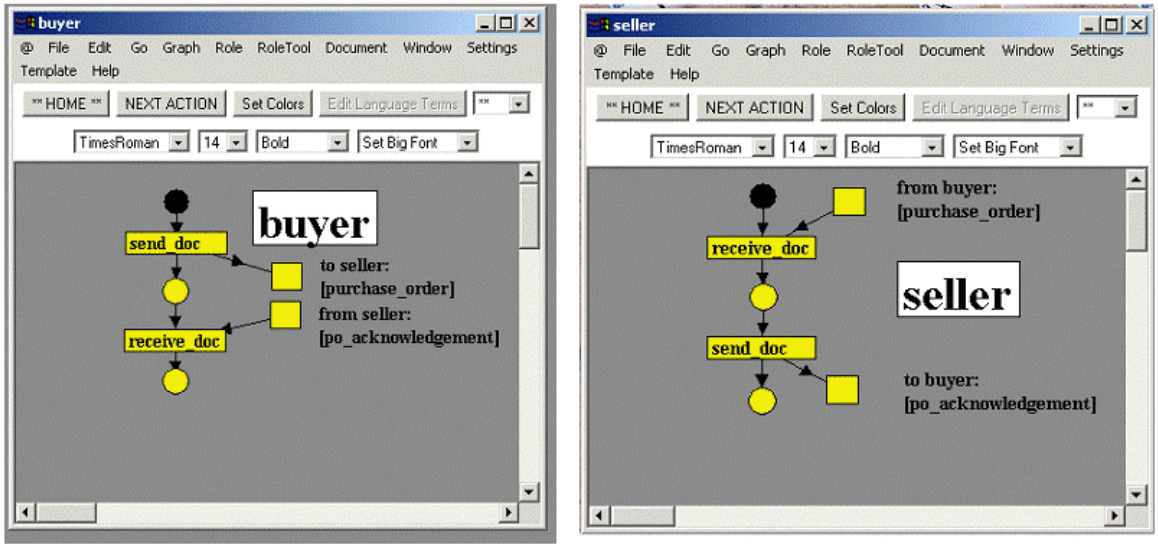


Figure 4.16 Screenshot of Role Procedures in InterProcs Executer.

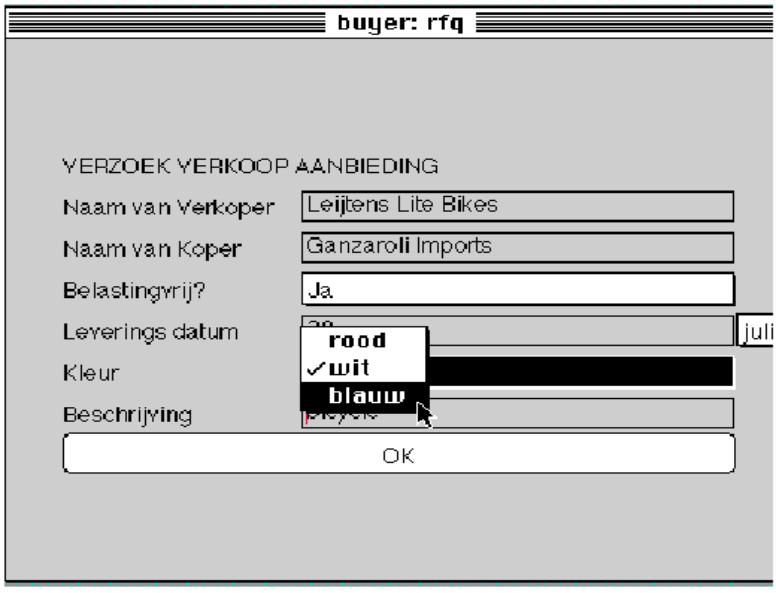


Figure 4.17 Sample Snapshot of EDI Document in Dutch.

At this stage, Professor Lee points out that the scenario produced are fixed, that means the trade scenarios can't be adapted or adjusted to meet the additional needs of a given situation. To address this problem, another concept called Procedure Constraint Grammar (PCG) was introduced by Professor Lee (Lee, 1999).

The objective of the PCG representation is to describe procedures by their temporal ordering constraints, rather than by the absolute sequence of steps. By this expert system approach, scenario components can be broken down into reusable component parts that can be flexibly reassembled to meet the needs of a wide variety of situations.

Professor Lee (Lee, 1999) suggests that the Procedure Constraint Grammar can be used in the following way: the user interacts with the system, specifies constraints and objectives of trade procedure. Based on these specifications, the system generates a trade scenario represented in a graphical form. This trade scenario then can be compiled and simulated. Work on Procedure Constraint Grammar is now in progress by Professor Lee and his research team.

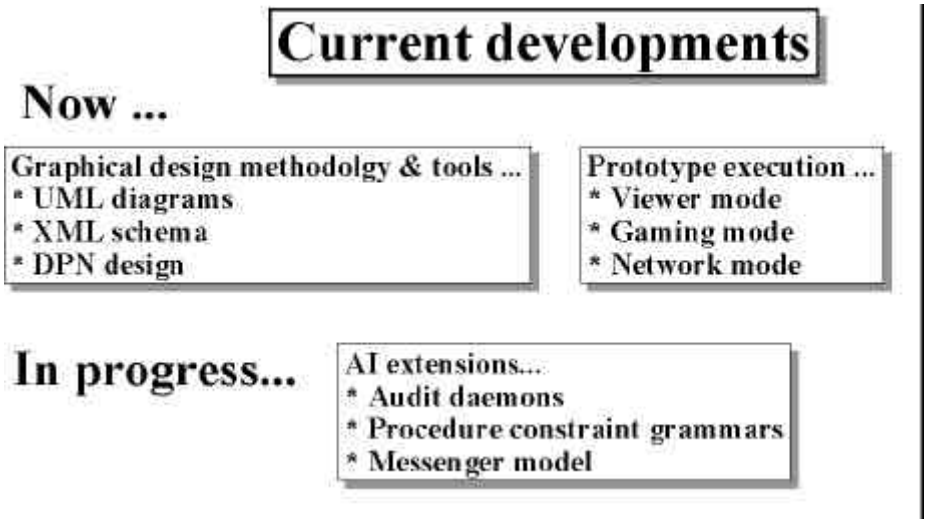


Figure 4.18 Current Development Phase of InterProcs.

The current development phase on InterProcs is given by Professor Lee, which is illustrated in Figure 4.18. We adopt this figure from (Lee, 1998). A principal feature of InterProcs is its graphical user interface for designing trade scenarios. It also offers the possibility of simulating these scenarios, including graphical animation of the transaction flow.

InterProcs is written in Java, the main value of the InterProcs is that it can help in the areas of designing and prototyping trade scenarios. Professor Lee points out (Lee, 1999) that the InterProcs has been developed in such a way as to be able to preserve the resulting trade scenario as object-oriented components. These components could be stored in a publicly accessible repository; trading partners involved in an international

trade procedure can download and executed the trade scenarios. According to Professor Lee, these components may be combined with other business system components like security, database access, and application interfaces to provide a production-level implementation. In this way, the mutual trust-related problem in the open electronic commerce environment can be solved effectively.

The final goal of InterProcs is illustrated in Figure 4.19. We redraw this figure from (Lee, 1999). The optimal vision involves scenarios designed with InterProcs to be made freely accessible as libraries from independent agencies via the Internet. These libraries, along with XML schemas, would be integrated with reusable component architecture such as Enterprise JavaBeans. Further more, this architecture would combine functionality from off-the-shelf and customer developed business applications.

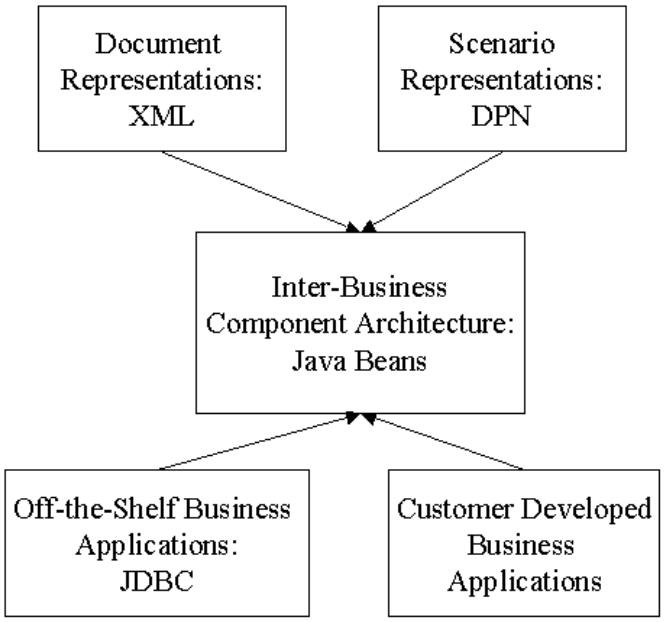


Figure 4.19 Final Goal of InterProcs.

Great efforts have been put on InterProcs by Professor Lee and his research team. The first version of InterProcs was released in June 2000. We experimented with their thirteenth public release InterProcs.22feb2001 in April 2001. We successfully created a simplified international trade scenario called GT3 using InterProcs.22ffb2001. There are 3 roles and 5 documents in GT3. This simple trade scenario will be discussed in detail later in this chapter and our implementation in the next chapter also will base on this simplified scenario.

Unfortunately, the InterProcs.22feb2001 was not very stable at that time. In May 2001, we wanted to make some changes to the GT3 model, such as adding a passing document to a role. At this stage InterProcs.22feb2001 usually gave some error messages and refused to carry on. Also when using the InterProcs built-in function to make a Role Procedure Graph from a Joint Procedure Graph or vice versa, we need to manually readjust the positions of documents and arcs to make them identifiable, otherwise they will mixed up together.

In Professor Lee's Documentary Petri Nets, the only way to communicate among trading parties is through documents, so the research team pays more attention to documents, such as the structure of the document, document content, different representation of document in multi-language. These efforts make InterProcs very powerful at dealing documents in design phase. But in InterProcs we can't find many simulation features such as collecting data, managing simulation time, etc. These simulation features in fact can be used to support the analyses of trade scenario in execution phase.

In our point of view, the lack of simulation features in InterProcs is due to two reasons. Firstly, Professor Lee intended to develop InterProcs as an application in product-level, not especially for experiment and analysis purpose. Secondly, InterProcs is programmed in Java, Java is a general purpose programming language, it does not provide support on those simulation features itself, and we were unable to find a full-featured Java library of simulation routines. Our web search revealed that dozens of programming teams have implemented special purpose simulation systems in Java, apparently in the attempt to construct a general-purpose simulation library. (This gap in Java support will undoubtedly be filled someday). Also in InterProcs, other parts in progress, such as audit daemons, and procedure constrain grammar, are important on production level.

In this section, we introduce Professor Lee's electronic trade scenario. The Documentary Petri Nets representation was presented as a candidate representation for such trade scenarios. The InterProcs system was presented as a prototyping environment to support the design and execution of such trade scenarios. It offers a graphical user interface for designing trade scenarios.

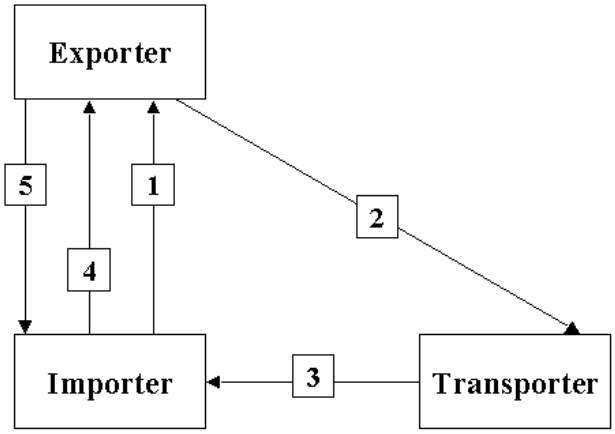
Professor Lee (Lee, 1992) did not give much detail on how to model and analyse business process for Documentary Credit Procedure in international trade. By simplifying Documentary Credit Procedure illustrated by Figure 2.1 in Chapter 2, we design a simple trade scenario called GT3. Then we will use the Business Process Reengineering techniques discussed in Chapter 3 to analysis and model GT3 to improve understanding of Business Process Reengineering in international trade. In section 4.2, we will design and analysis GT3.

4.2 Simplified Trade Scenario GT3

In this simple trade scenario, we have three parties involved. They are Importer, Exporter, and Transporter. Through this simple trade scenario, the way of doing business among all the participants is well known by the three parties. As defined in Figure 4.1, the GT3 scenario only deals with the negotiation phase, which is after the Importer has identified the prospective Exporter for the products or services to be purchased. Communication between these three parties is via the documents.

The trading procedure is initiated by Importer sending a Purchase_order to the Exporter, that is the first step in this trade procedure. In the second step, after processing this Purchase_order, Exporter sends Goods delivery instruction to Transporter. In the third step, when Goods are ready, Transporter sends Goods to Importer. In the fourth step, after Importer receives the Goods, it arranges to send Payment to Exporter. In the fifth step, when Exporter receives the Payment, it sends a Payment_acknowledgement to Importer. Figure 4.20 illustrates the whole trade scenario. Labelling numbers indicate the ordered steps in this trade procedure, and the list at the bottom identifies the names of the documents in each step.

Here we start analysis of the workflow in GT3 with the specification of the communication structure. In GT3 the involving parties interact with each other via documents only, each party has its own private workflow; there is no need to agree upon one common process for the internal workflow. But for interaction purpose, they must agree upon a protocol, in our case this is the trade scenario, such as the structure of the document, encryption technology, etc. Here we are interested in the interface of the local workflow instead of the internal behaviour. It is very convenient for us to use Message Sequence Chart to get an overview of the message sequence in the system.



1. Purchase_order
2. Good_delivery
3. Goods
4. Payment
5. Payment_acknowledgment

Figure 4.20 GT3 -- A Simplified Trade Scenario for International Trade.

Figure 4.21 describes the interactions among the three parties involved in terms of a Message Sequence Chart. This Message Sequence Chart specifies the documents that are exchanged and the ordering of events associated with sending and receiving documents. The representation of Message Sequence Charts is intuitive and it focuses on the messages between communication parties. In GT3, we have five documents. Each document has a sender and a receiver. For example, the instance Importer is the sender of Purchase_order and the instance Exporter is the receiver of Purchase_Order. According to Mauw (Mauw & Reniers, 1994), an instance is an abstract entity of which one can observe the interaction with other instances in a Message Sequence Chart representation.

An instance is denoted by a vertical axis. The time along each axis runs from top to bottom. So we can clearly represent the document order. In Figure 4.21, the instance Transporter sends Goods only after the receipt of Goods_Delivery order from instance Exporter.

Our Message Sequence Chart of GT3 is the same as the second step of Processor Lee's methodology shown in Figure 4.8. Professor Lee calls it Sequence Diagram. We think it is necessary to follow this specific step in Professor Lee's methodology to depict the communication protocol using Message Sequence Chart, because all roles in GT3 communicate with each other via ordered document only.

After getting an overlook of the communication structure in GT3, we need now to deal with more detail to get a better understanding of this trade scenario, and obtain more details on each party. At this stage we follow the third step in Processor Lee's methodology. Professor Lee calls it Activities Diagram shown in Figure 4.8. We analysis this scenario using a Swimlane Diagram. The Swimlane Diagram is a specific modelling technique in workflow management designed to support the definition and control of processes. It can illustrate an entire business process from the beginning to the end (Sharp & McDermott, 2001) shown in Figure 4.22.

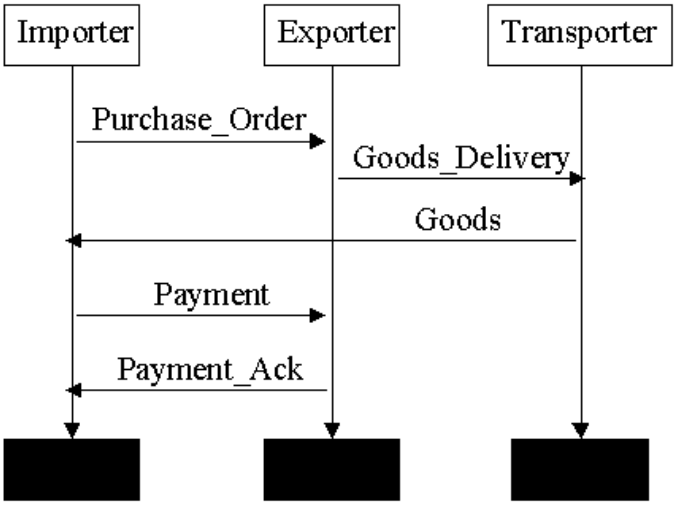


Figure 4.21 GT3 in Terms of a Message Sequence Chart.

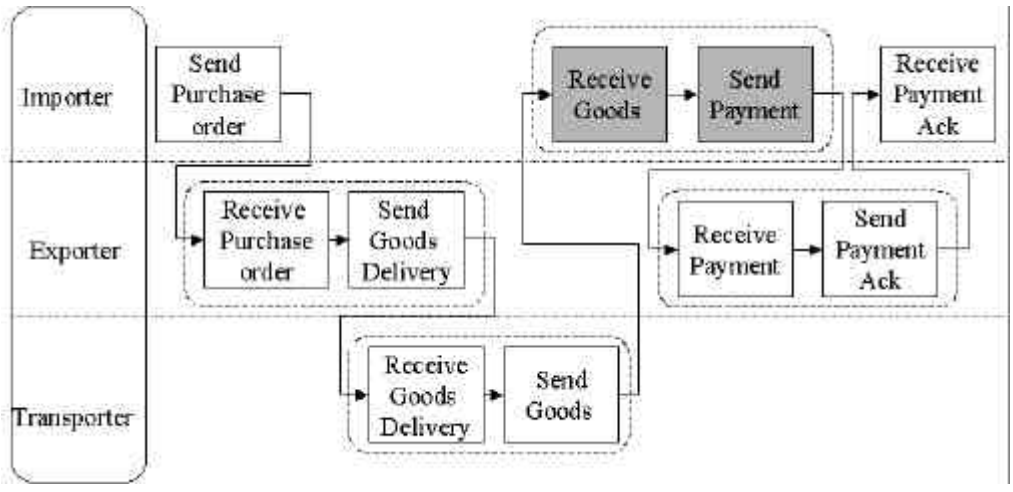


Figure 4.22 GT3 in Terms of a Swimlane Diagram.

In this section we design a trade scenario called GT3 by simplifying Documentary Credit Procedure for international trade. Also we identify two necessary steps in Professor Lee's methodology to analysis our GT3 model. In the next section we use Linear Documentary Petri Nets to represent our GT3 model.

4.3 Linear Documentary Petri Nets

In Professor Lee's Documentary Petri Nets, transitions are labelled to interpret the actions it performs; a new kind of place node called document place is added to represent the documents, each document place has an associated label to identify the name, sender, and receiver of the document; goods and currency can be treated as a special kind of document with the same representation form.

After making these additions, Professor Lee (Lee, 1999) indicates that the resulting Documentary Petri Nets are quite suitable to be used as a representation language for modelling electronic trade scenario.

Here we make three modifications on Professor Lee's Documentary Petri Nets to suit our needs. The resulting Documentary Petri Net is called Linear Documentary Petri Net.

- First Modification: Eliminating branches in Documentary Petri Net to concentrate on the most important case.

In our study of GT3, we emphasis our interest in analysing the most important case only. It means, for example, when Exporter receives the Purchase_Order, it will send Goods_Delivery instruction to Transporter. We do not consider the choices that a role can make and exceptions that may occur in the delivering of documents. Each document goes to the right recipient in the right order. For our purpose of demonstration, we think the time sequence is important in our trade scenario. Payment should be sent to Exporter after Importer receives Goods from Transporter. Sending Payment to Exporter on a specific time point is meaningless. After this modification, we obtain Documentary Petri Nets without branches.

- Second Modification: The maximum number of document that can be sent is one in a single documentary communication.

In Professor Lee's Documentary Petri Nets, sending bundles of documents in a single documentary communication is allowed. In order to make the Documentary Petri Nets more structural, more suitable for prototyping, we limited the number of document that can be sent in a single documentary communication to one. That means each transition can have at most one input document place, and at most one output document place as well.

This also greatly simplifies analysis. A node that receives, say, 4 documents has 2^4 possible input triggers (each of the 4 document maybe either present or absent). We will insist, when capturing scenarios from experts, that they identify a single "most important" case for analyses.

- Third Modification: Document labelling.

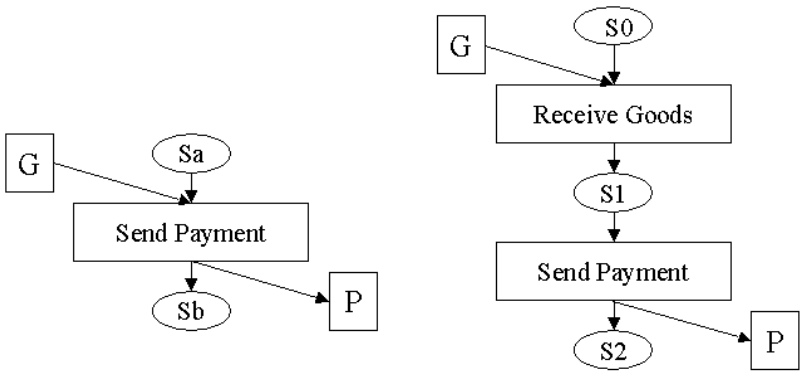
In Documentary Petri Nets, each document place has an associated label besides it. The label includes name, sender, and receiver of the document. This is only suitable for trade scenario without many documents. We change the labelling style as: "label the document place with the abbreviation of document name and move all other document attributes to the bottom of each role procedure". When we refer to document attributes, we mean the sender, receiver of the document. This makes the graph clean and tidy and also manageable for prototyping.

We name the modified Documentary Petri Nets as Linear Documentary Petri Nets (LPDN). Linear Documentary Petri Net does not have a branch; it limits the number of documents that can be sent to one in a single documentary communication; document places are labelled with the abbreviation of the documentary only, the other attributes of the documents are moved to the bottom of each role procedure. This resulting Linear Documentary Petri Nets preserve all the characteristics of Professor Lee's Documentary Petri Nets. It is more structural and easy to manage for demonstration purpose.

We use Importer procedure as an example to illustrate how to represent it into a Linear Documentary Petri Net. We can easily identify the actions performed by the Importer from Figure 4.22. It performs the first action to send a Purchase_order to Exporter to initiate the whole trade procedure; after receiving Goods, it performs the second action to send Payment to Exporter; finally when the Exporter sends Payment_Ack back, Importer will perform the third action to receive it.

Actually Importer has performed two actions in the second step, first it receives Goods, then sends Payment to Exporter. This is illustrated in Figure 4.23. The point here is in studying this trade scenario, our emphasize is on the communications between parties. The workflow within a particular party is of less interest.

Our assumption is each party involved in this trade scenario is well behaved. A well-behaved Importer will send Payment after it receives Goods delivered by Transporter. As described in Figure 4.23 (b), there is no document output for transition "Receive Goods", no document input for transition "Send Payment". State S1 can be reached if both state S0 and document place G are marked; if state S1 is marked then document place P will be marked and S2 can be reached automatically. So state S1 can be bypassed and these two actions can be combined together. Therefore, Figure 4.23 (a) has the same effect as Figure 4.23 (b) for our research purpose.



Document	Name	Sender	Receiver
G	Goods	Transporter	Importer
P	Payment	Importer	Exporter

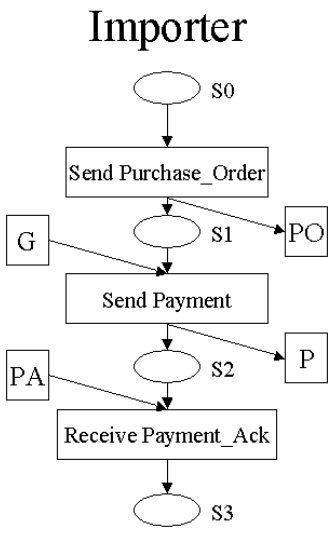
Figure 4.23 Illustration of Actions Combining.

Also for a particular party, we think sending action is more active than receiving action. That is the reason why we use label "Send Payment" instead of "Receive Goods and Send Payment". An advantage of this labelling style lies in the simplification of action labelling and this makes the graph clean and tidy.

This combination of actions can also be illustrated with our previous Swimlane diagram representation shown in Figure 4.22. In Importer Swimlane, the two shadowed boxes, Receive Goods and Send Payment, are all performed by the same role -- Importer.

Because our research focus is on interactions between parties instead of within a particular party, so handoffs -- special kind of flow that cross the Swimlane from one role to another, are our focus. Therefore we can combine these two actions in the Swimlane to abstract our representation to a higher level. These simplifications make the model understandable to the most people with the least effort. In Figure 4.22, we also identify other actions that can be combined together which are all shown in the dash line boxes.

Linear Documentary Petri Net representations of each role involved in our simple trade scenario are illustrated through Figure 4.24 to Figure 4.26. In Importer Linear Documentary Petri Net shown in Figure 4.24, whenever an Importer instance is generated by the system, Importer is initiated at state S0. It firstly performs the action Send Purchase_Order, this newly produced Purchase_Order is represented as a document place called PO as the output of this action, and then it stops to wait for the Goods to be delivered by Transporter. At this stage, state S1 is reached.

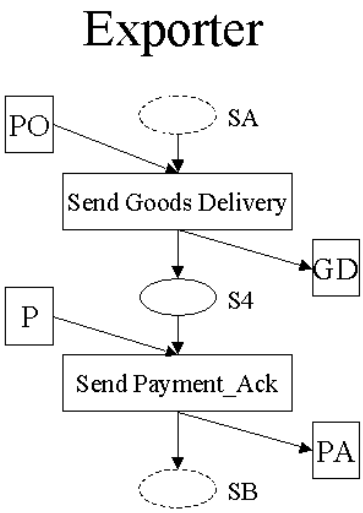


Document	Name	Sender	Receiver
PO	Purchase_Order	Importer	Exporter
G	Goods	Transporter	Importer
P	Payment	Importer	Exporter
PA	Payment_Ack	Exporter	Importer

Figure 4.24 Importer Procedure Represented in Linear Documentary Petri Net.

When Importer receives Goods, it performs the second action Send Payment to Exporter. So the receiving Goods is represented as an input document place labelled as

GO, and the sending Payment is represented as an output document place called P of this action. After this transition, state S2 is reached. When the Payment_Ack from Exporter arrives, Importer performs its last action Receiving Payment_Ack, and reaches its final state S3. This receiving of Payment_Ack is represented as an input document place called PA of this action.



Document	Name	Sender	Receiver
PO	Purchase_Order	Importer	Exporter
GD	Goods_Delivery	Exporter	Transporter
P	Payment	Importer	Exporter
PA	Payment_Ack	Exporter	Importer

Figure 4.25 Exporter Procedure Represented in Linear Documentary Petri Net.

In the same way we can get Linear Documentary Petri Net representation of the other two roles shown in Figure 4.25 and Figure 4.26. Note that in Importer workflow, we have an input state S0 and an output place S3 for our trade scenario GT3, but for Exporter and Transporter, there is no need for these state places. Because workflow in Exporter is initialised by the passing document PO from Importer only, therefore source place SA in Figure 4.25 can be omitted, so is SC shown in Figure 4.26. However, for semantically reasons we assume that there is one source place and one sink place for Exporter and Transporter procedure. These source and sink places are represented as dashed circles called SA, SB, SC, and SD respectively, which are shown in Figure 4.25 and Figure 4.26.

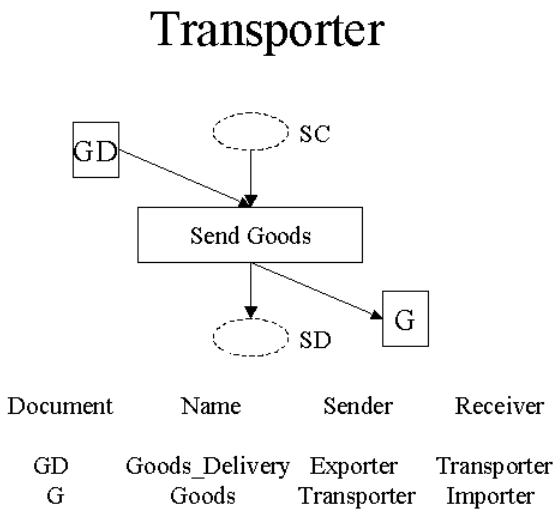


Figure 4.26 Transporter Procedure Represented in Linear Documentary Petri Net.

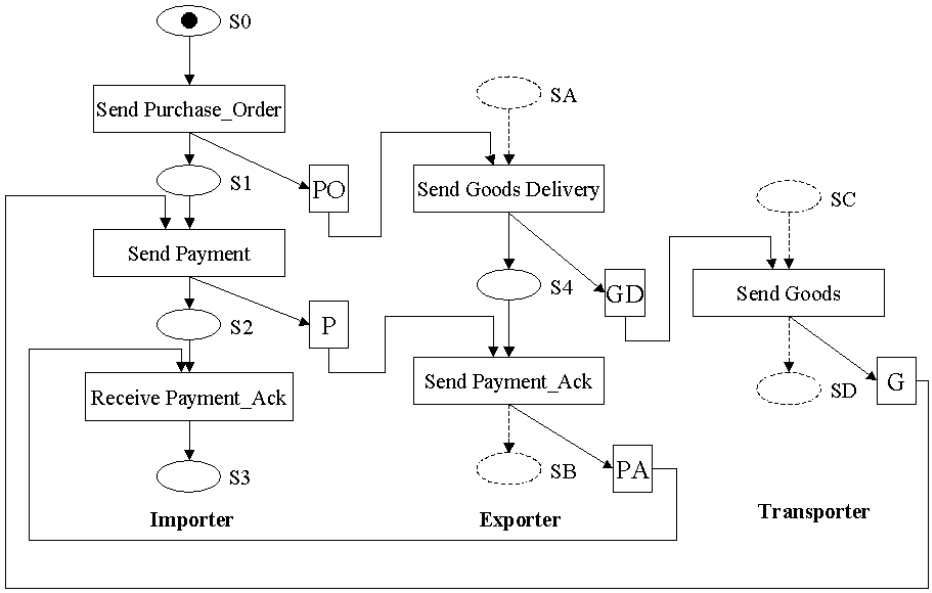


Figure 4.27 GT3 Execution Demonstration Step 1.

Finally, we will demonstrate how GT3 can be executed in terms of Petri Nets to obtain an overview of the whole procedure in a particular trading case. These are illustrated through Figure 4.27 to Figure 4.33.

In Figure 4.27, when there is a black token in state place S0, transition Send Purchase_Order is fired, this black token is consumed and two black tokens are produced and put in state place S1 and document place PO. This is illustrated in Figure 4.28. At this stage, transition Send Payment is enabled, but it needs to wait for a black

token in its input place G to be fired. While transition Send Goods Delivery is fired, because it has only one input place PO and there is a black token in it.

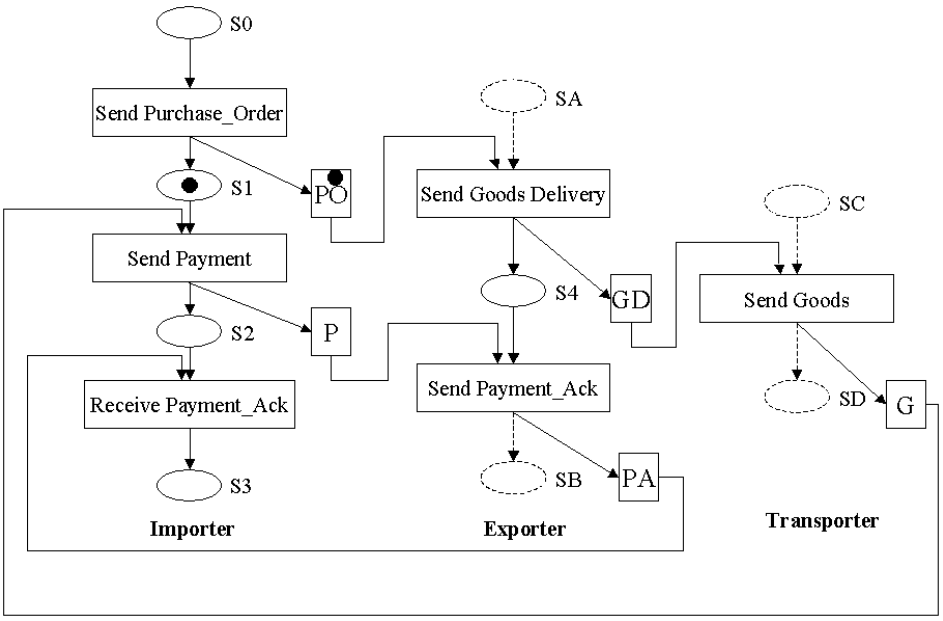


Figure 4.28 GT3 Execution Demonstration Step 2.

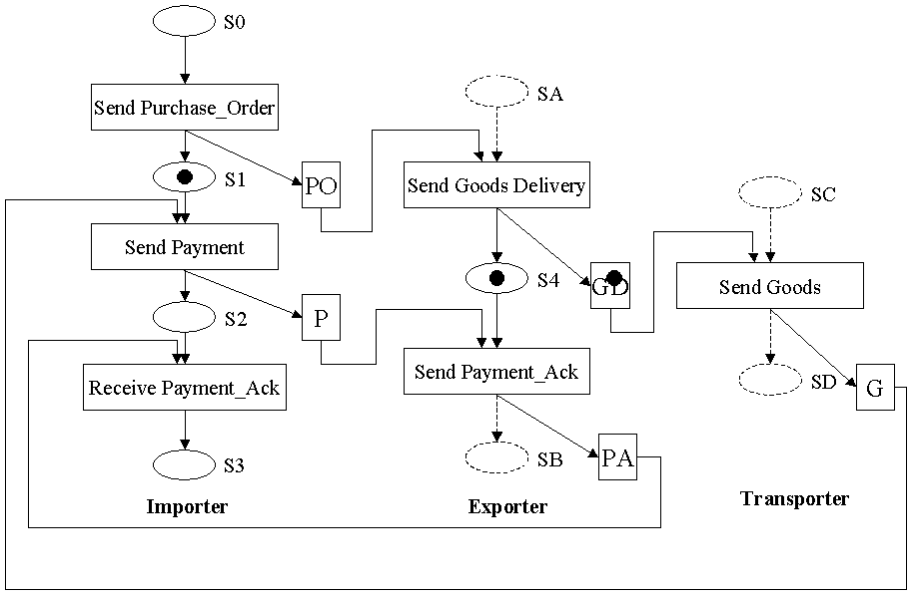


Figure 4.29 GT3 Execution Demonstration Step 3.

Figure 4.29 illustrates the firing of transition Send Goods Delivery. Black token in document place PO is consumed; two black tokens are produced and put in state place S4 and document place GD. At this moment, transition Send Payment_Ack is enabled, but it needs to wait for another black token in document place P to be fired. Transition

Send Goods is fired because there is a black token in its only input place GD. This process is shown in Figure 4.30.

In Figure 4.30, transition Send Goods consumes the black token in document place GD and produces a black token in G. At this time, transition Send Payment is fired because there are tokens in both its input places G and S1. These two tokens are consumed and two new black tokens are produced and put in state place S2 and document place P. At this stage, transition Receive Payment_Ack is enabled, but it needs to wait for a black token in another input place PA. Transition Send Payment_Ack is fired by the black token in document place P and the previous generated black token in state place S4. This procedure is illustrated in Figure 4.31.

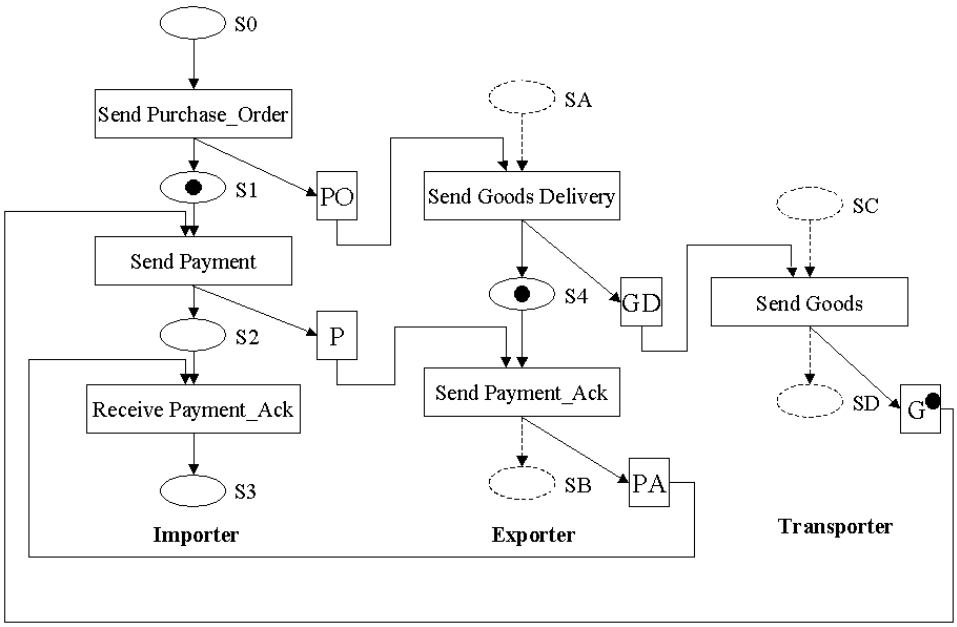


Figure 4.30 GT3 Execution Demonstration Step 4.

In Figure 4.31, transition Send Payment_Ack consumes black tokens in document place P and state place S4, a new black token is produced in document place PA. This black token with the previously generated black token in state place S2 fires transition Receive Payment_Ack shown in Figure 4.32. Black token in S2 and PA are consumed by Receive Payment_Ack, a new black token is generated in state place S3 shown in Figure 4.33. This state place S3 is the sink state of the trade scenario represented by Linear Documentary Petri nets.

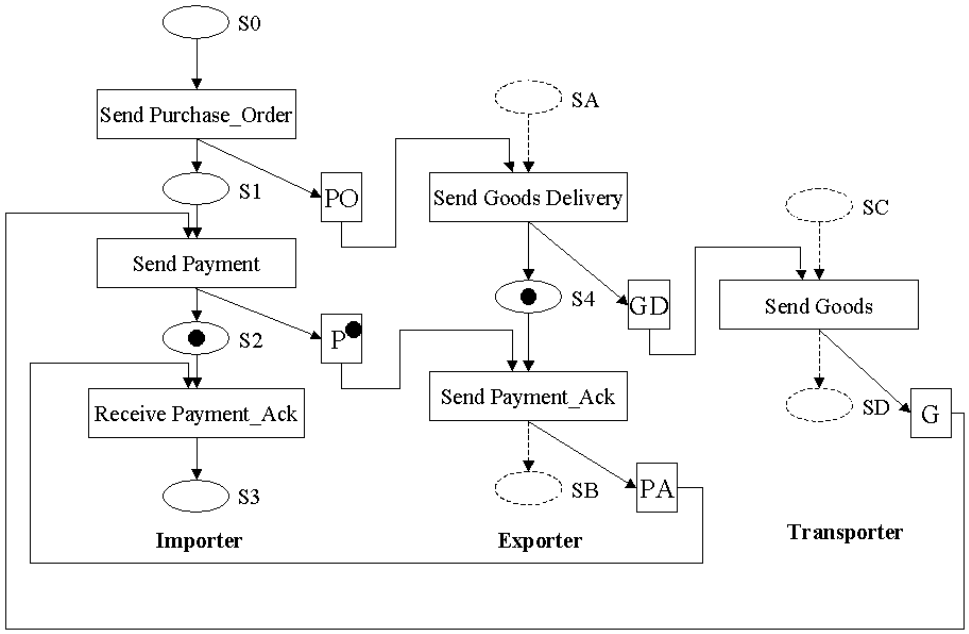


Figure 4.31 GT3 Execution Demonstration Step 5.

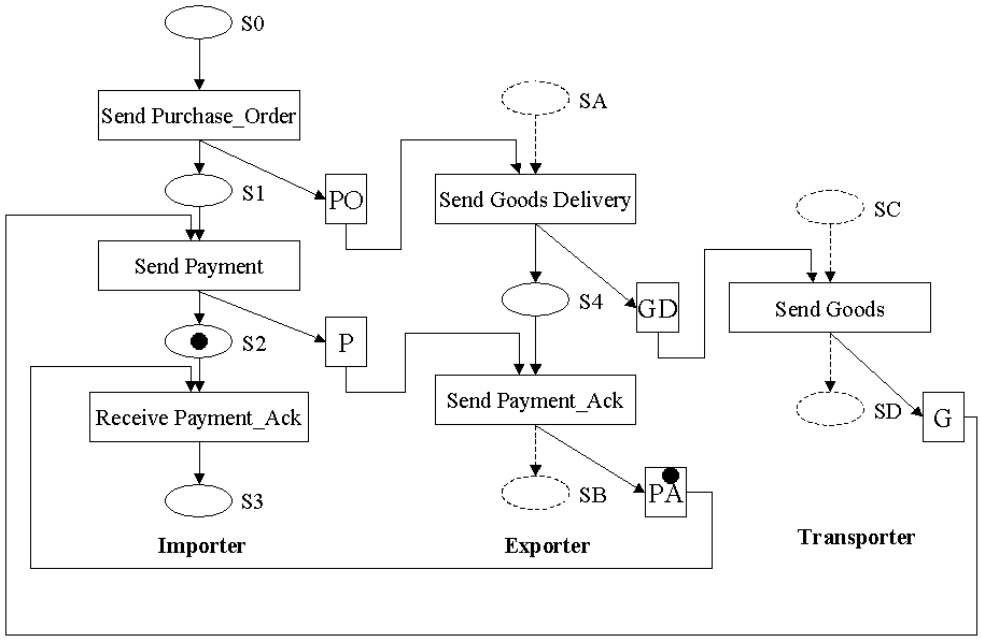


Figure 4.32 GT3 Execution Demonstration Step 6.

In this section we modified Professor Lee's Documentary Petri Nets by getting rid of branches; limiting the number of documents that can be sent to one in a single documentary communication; rearranging the way that document places are labelled. For representing electronic trade scenario, the resulting Linear Documentary Petri Nets are more structural and easy to manage besides preserving all the characteristics that Documentary Petri Net has.

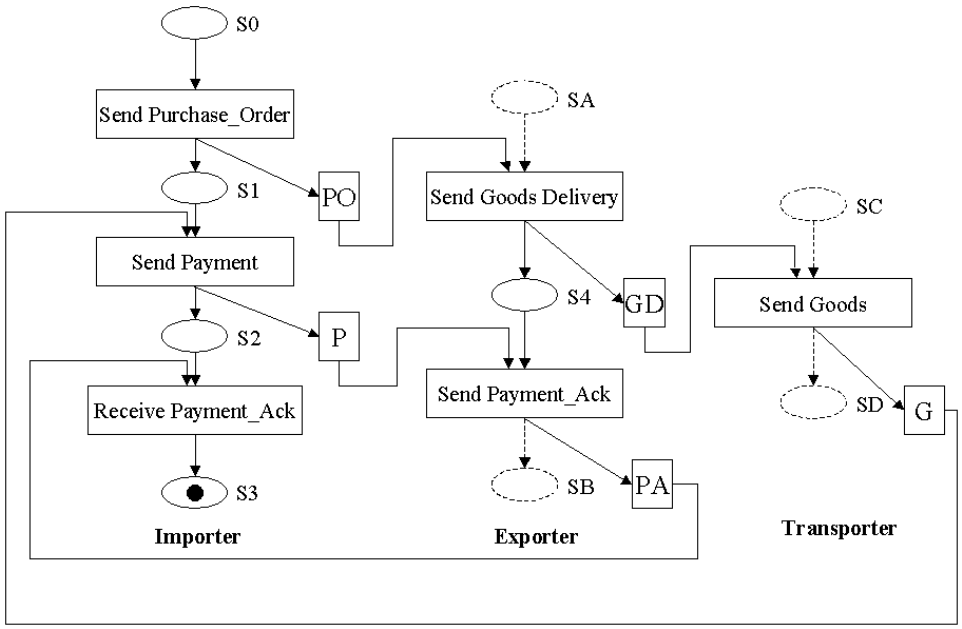


Figure 4.33 GT3 Execution Demonstration Step 7.

4.4 Summary

In this chapter, we discuss Professor Lee's work on reengineering international trade in detail. It is believed by Professor Lee that electronic trade scenario can be a solution to deal with the mutual trust-related problems in international trade within open electronic commerce environment. These trade scenarios are the mutually agreed-upon set of procedures, documents, and rules that control the activities among all the parties involved. Electronic trade scenario is represented by Documentary Petri Nets. A production-level case tool called InterProcs is also made available by Professor Lee to assist in designing and prototyping trade scenarios.

We design a simple trade scenario called GT3 by simplifying the Documentary Credit Procedure for international trade. We also identify two necessary steps in Professor Lee's methodology and use several techniques in Business Process Reengineering to analyse GT3 for better understanding of it. Using these techniques together help us to obtain useful knowledge from different angle and level of abstraction. Later in this chapter, we use a modify Documentary Petri Nets called Linear Documentary Petri Nets to represent our GT3 model for further study.

Chapter 5 Simulation of GT3

In the previous chapter, we discussed the simple trade scenario GT3 and represented it with Linear Documentary Petri Nets. In this chapter, we intend to describe the simulator we wrote to simulate the workflows in GT3 to demonstrate how the system works. Then we try to reach a more ambitious goal, that is, to produce a tool with which non-technical users can design a new trade scenario or modify the existing model easily for training purpose. In this way, users can get better understanding of the system under study without much special knowledge on programming.

We start by identifying that simulation is a powerful tool to help people obtain better understanding of the system under study. Then we discuss the choice of simulation model and programming language. After that we describe our implementation of GT3. At last we make attempt to produce a tool for helping users to generate a new trade scenario automatically based on the simple specifications that users provide. This newly generated trade scenario can be executed automatically or manually.

5.1 What is Simulation?

According to Davies and O'Keefe (Davies & O'Keefe, 1989), when the word "simulation" is used by computer scientists, statisticians, and management scientists, they normally refer to the construction of an abstract model representing some system in the real world.

> The simulation describes the pertinent aspects of the system as a series of equations and relationships, normally embedded in a computer program.

Johnston (Johnston, 1987) describes a computer simulation as:

> ...A computer program that defines the variables of a system, the range of values those variables may take on, and their interrelations in enough detail for the system to be set in motion to generate some output. The main function of a computer simulation is to explore the

> properties and implications of a system that is too complex for logical or mathematical analysis... A computer simulation generally has an ad hoc or 'home-made' quality that makes it less rigorous than a mathematical model

These descriptions might be adequate for many purposes. The basic idea behind simulations is simple. When we want to get knowledge or make decision on a real world system, usually the system is not easy to study directly due to its complexity; we then proceed indirectly through creating and studying another system (simulation model), which is sufficiently similar to the real world system. And we are confident that what we learn about the model is also true of the real world system.

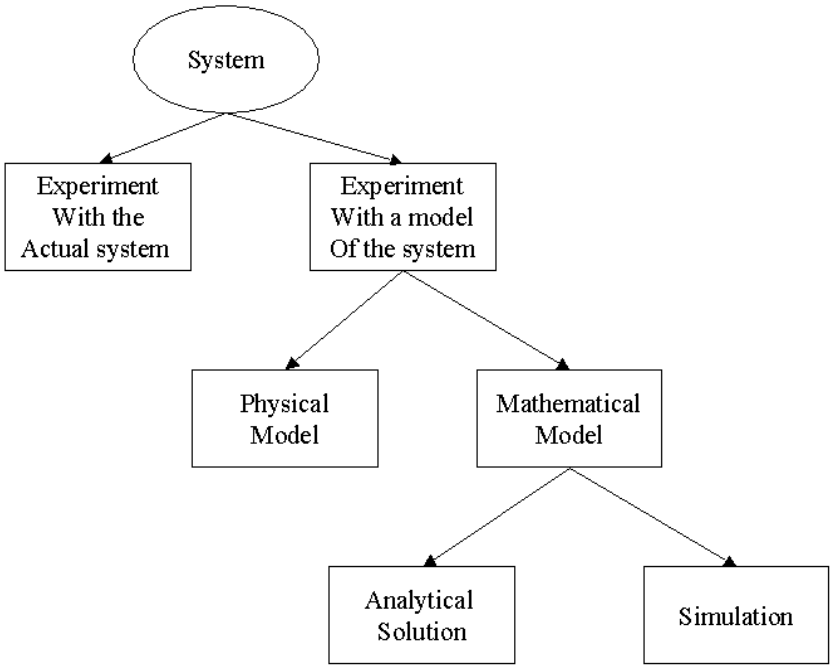


Figure 5.1 Ways to Study a System.

Figure 5.1 illustrates different ways that a system can be studied. We redraw this figure from (Law & Kelton, 1991). The authors give detailed explanations on their methods of studying a system.

- Experiment with the actual system vs. Experiment with a model of the system.

If it is possible to make changes to the actual system, and let it operate in the new conditions, then it is desirable to do so. However, in real world, it is rarely feasible to do so. Law and Kelton (Law & Kelton, 1991) give the simulation of a non-existing system (a system not being built yet) as an example, to study the proposed alternative

configurations to see how it should be built. Another example they give is to study the simulation of an existing bank system; it is deduced that reducing the tellers in actual system in order to decrease the cost, could lead to a long service delay to customers. So it is necessary to build a model to represent the system being studied, because the experiments with the real system are too costly or too disruptive.

- Physical model vs. Mathematical model.

According to Law and Kelton (Law & Kelton, 1991), physical models are suitable for engineering or management systems. For example, tabletop scale models of material-handling system. For representing a system in terms of logic and quantitative relationships, to see how the system reacts due to manipulation and changes being made to the system, mathematical model is more appropriate.

- Analytical solution vs. Simulation.

After setting up a mathematical model, it must be examined to see how it can be used to study the system being represented. For a simple model, the authors think that it may be possible to work with its relations and quantities to get an exact, analytical solution. If an analytical solution to a mathematical model is available and is computationally efficient, then analytical solution is the desirable way to study the system. But the real world systems are usually very complex, even the mathematical models of them are complex too. Simulation, at this stage, is regarded as a better way for studying them.

Through the above discussion, we give the definition of simulation, and talk about some different ways of studying a system. The purpose of systems study is for better understanding of the relationship among various components and for predicting performance when changes are made to the systems. Compared with other methods, simulation is more cost-effective, timesaving, and safer. Law et al (Law & Kelton, 1991) list some application areas in which simulation has been regarded as useful and powerful tool. Some of these areas are:

- Designing and analysing manufacturing systems.
- Designing communications systems and message protocol for them.
- Analysing financial or economic systems.

- Designing and operating transportation facilities such as freeways, airports, subways, or ports.

In real world, no two simulation projects will be identical, but some generalization can be made in the simulation projects. Pidd (Pidd, 1988) indicates that simulation work can be viewed as having three phases as shown in Figure 5.2.

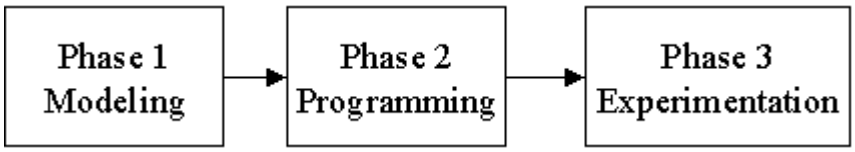


Figure 5.2 Pidd's Three Phases for Simulation Work.

Pidd (Pidd, 1988) also states that the three phases are very difficult to separate precisely in practice. Because it is very difficult or impossible to program properly without an adequate model, so some overlaps will occur among these three phases in the whole simulation process.

In this section we discussed the concept of simulation and its advantages over other methods on studying a real world system. Also we identified the three phases defined by Pidd (Pidd, 1988) for typical simulation work. In the next two sections, we will discuss the first two of these three phases in detail, they are: Modelling and Programming.

5.2 Choice of Modelling Techniques

For a simulation model, the most important issue is the principal elements. So the simulation analyst must decide what his principal element is for his specific case. For this purpose, Pidd (Pidd, 1988) states that two aspects should be taken into consideration. One is the "nature of the system being simulated", because some modelling approaches are more suited to certain problems than to others. Another is the "nature of the study being carried out", which refers to what the object of the study is.

When both of these two aspects are taken into consideration, the analyst will get a clear idea on his simulation project, so that he can decide what kind of level of accuracy and detail are appropriate for the simulation being proceeded.

Pidd (Pidd, 1988) categorizes simulation models into three groups: time-slicing or next event; deterministic or stochastic simulation models; and continuous or discrete simulation models. In Law et al's (Law & Kelton, 1991) categorization, the first category: time-slicing or next event, is replaced by static or dynamic simulation models. The other two categories are the same as Pidd's. Graybeal and Pooch (Graybeal & Pooch, 1980) also have the similar categorization of simulation models. We will give a brief overview on these simulation models.

- Time slicing vs. Next event models.

In this simulation model, Pidd (Pidd, 1988) indicates that state changes of the system are modelled through time. To consider how time flow can be handled within simulation is important.

Time slicing approach involves updating and examining the model at regular time intervals. So for a time slice of length dt, the model is updated at time (t + dt) for changes in the time interval (t to (t + dt)). The obvious problem for this approach is that whether there are any events to carry out or not, the model must be advanced by a time slice. If the time slice is too large, then it is impossible to simulate some of the state changes that occur.

Instead of regular time interval, the next event technique uses variable time increment. Model is only examined and updated when it is known that a state change is due or the next event to be executed, regardless of the time interval. This method is more efficient than time slicing, especially where events are infrequent.

- Static vs. Dynamic simulation models.

Law et al (Law & Kelton, 1991) state that a static simulation model is a representation of a system at a particular time; examples of static simulations are Monte Carlo models. In Monte Carlo models, random values are selected to simulate a model. It's the same with the variables that have a known range of values but an uncertain value for any particular time or event. On the other hand, a dynamic simulation model represents a system as it evolves over time, such as a conveyor system in a factory.

- Deterministic vs. Stochastic simulation models.

From Graybeal and Pooch's point of view (Graybeal & Pooch, 1980), a deterministic system is a system whose behaviour is entirely predictable. In a deterministic system, the new state of the system is completely determined by the previous state and by the activity. That means the system is well understood, and it is possible to predict precisely what will happen. A cycle of operations on an automatic machine is a good example. The authors give two figures to illustrate these two systems corresponding to deterministic and stochastic simulation modes.

A stochastic system is one whose behaviour cannot be entirely predicted, the system contains a certain amount of randomness in the state transitions. In some case it might not be possible to assign a probability to the state that the system will assume after a given state and activity.

In Figure 5.3 (a), a deterministic system is depicted. S_0 refers to the state of the system before activity A. S_n refers to the state of the system after the activity A. In Figure 5.3 (b), S_1 and S_2 are two possible states that the system can enter after the state S_0 in response to activity A. So a stochastic system is nondeterministic because the next state cannot be unambiguously predicted if the present state and the activity are known. We redraw this figure from (Graybeal & Pooch, 1980).

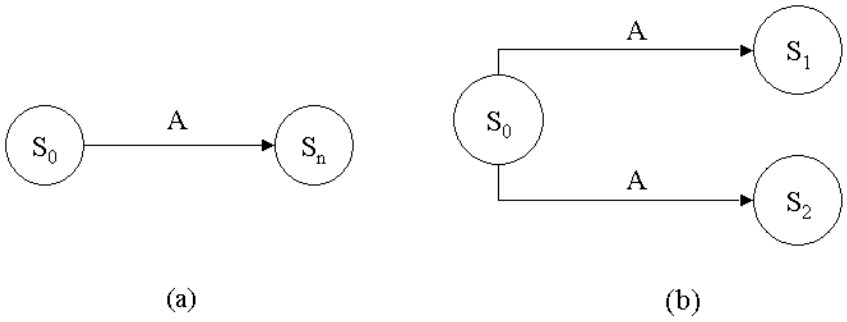


Figure 5.3 A Deterministic vs. a Stochastic System.

Pidd (Pidd, 1988) also gives an example to show that the difference between a deterministic and stochastic system. A lecturer may give the same lectures to several sets of students, but the duration may vary. Although we can say that the lecture is normally distributed with a mean of 50 minutes and a standard deviation of 3 minutes by statistic statements, but it is impossible to state precisely how long a particular delivery of the lecture will last unless the lecturer's behaviour can be completely controlled, and that of the class as well. So Pidd thinks that the distinction between

deterministic and stochastic systems is artificial. The distinction is highly based on the amount of knowledge about a system and the amount of control over that system studied by an observer.

- Continuous vs. Discrete simulation models.

The terms continuous and discrete applied to a system refer to the nature or behaviour of changes with respect to time in the system state. Systems whose changes in state occur continuously over time are continuous systems; systems whose changes occur in finite quanta or jumps, are discrete systems.

Consider a train moving from station to station, picking up and depositing passengers at each stop. In Pidd's (Pidd, 1988) point of view, there are some obvious system events are discrete change, such as 'train stop', 'door open', 'door close', etc. Thus, to simulate this system using a discrete model is desirable.

On the other hand, when the electronically powered locomotive moves from station to station, its speed will increase smoothly from the rest until it reaches an appropriate cruising rate. The speed doesn't change by discrete amount. Thus if the results of the simulation are to include the state of the system in relation to the continuous variable 'speed', then a continuous model is more suitable than a discrete model.

Sommerville (Sommerville, 1992) states that there are a large number of potential areas for discrete event simulation. One of the main areas currently being explored is in designing new manufacturing areas, especially where high capital investment is involved. For example, if a company wishes to build a new production line, then the line can be first simulated to assess feasibility and efficiency.

In the rest of this section, we will spend some time on discrete event simulation. Discrete event simulation describes a system in terms of logical relationships that cause changes of state at discrete points in time rather than continuously over time. Pidd (Pidd, 1988) indicates that quite a variety of systems can be regarded as having a queuing structure and as such they lend themselves well to discrete event simulation. Typical problems in this area are: Objects (customers in a gas station, aircraft on a runway, and jobs in a computer) arrive and change the state of the system instantaneously.

A discrete event simulation employs a next event technique to control the behaviours of the system, as its name indicates. Discrete event simulation concerns the modelling of a system as it evolves over time by a representation in which the state variables change instantaneously at separate points in time. Law et al (Law & Kelton, 1991) point out that due to discrete event simulation's characteristics, it is suitable for most of the real world system simulations.

Discrete event simulation is quite desirable to model our trade scenario for the following reasons. In our trade scenario GT3, system state evolves along the way from the start of a business case to the end of this particular case, changes at discrete point of time. For example, when the Importer sends out a Purchase_order to the Exporter, state of the system changes, so does with other events like the Transporter sends Goods to the Importer, the Exporter sends Payment_Ack to the Importer, etc.

Graybeal et al (Graybeal & Pooch, 1980) introduce some basic terminology in discrete event simulation. The objects of the simulation model are called entities, which are characterized by a fixed collection of parameters called attributes. Sets are collections of individual entities having common properties. The state of the system at any given time is completed described by the current list of individual entities, their attributes and the set memberships.

An event is an instant of time at which a significant state change occurs in the system. Such as an entity enters or leaves a state. Entities move from set to set because of the operations in which they involved. The operations and procedures, which are initiated at each event, are known as activities.

Temporary entities are created and destroyed during the course of the simulation, while permanent entities remain through the simulation. Sometime it is useful to group together a sequence of events in the chronological order in which they will occur. Such a sequence is known as process and is often used to represent all or part of the life of temporary entities.

In this section, we gave an overview on the issues of modelling – Pidd's first phase to simulate a real world system, see Figure 5.2. We also introduced some terminology for discrete event simulation. We argued that discrete event simulation is quite suitable for

modelling our trade scenario because the state of the system changes at discrete point of time while events occur.

In the next two sections, we will discuss the second phase of our simulation of GT3: Programming. We discuss the Programming phase in two steps. First we choose a implementation language in section 5.3, then in section 5.4, we discuss the implementation of GT3 in our chosen language.

5.3 Programming Language Issues

Techniques involved in this phase are program design and the choice of appropriate programming languages. Pidd (Pidd, 1988) states that whatever the choice of programming language, there is a growing tendency for a highly disciplined and structured approach to be taken to the programming. This is particularly important in large or complex programs.

Once the modelling approach has been decided, next thing to do is to translate the model into a form suitable for computer. That is, the model must be coded using some language.

The programming phase seems one of the best understanding steps in mode building, but it is very difficult sometimes to choose a suitable programming language. The range of programming languages that have been used for discrete event simulation covers the entire spectrum, from low-level machine-oriented assembly language to the specialized simulation-oriented language.

5.3.1 Common Factors in Language for Simulation

Graybeal et al (Graybeal & Pooch, 1980) discuss the languages issues. They identify seven factors that can influence the choice of a language for simulation. The seven factors are: Generating random varieties, Managing simulation time, Handling routines to simulate event executions, Managing queues, Collecting data, Summarizing and analysing data, and Formulating and printing output. Below are quotes from their prescription of these factors.

> (Generating random varieties)…to simulate probabilistic event usually requires the use of sampling techniques…then any language to be used in implementing a simulation model should either provide

> or allow easy development of a facility to generate and transform standard uniform random varieties.
>
> (Managing simulation time)…some means of representing time is required. This is important not only for controlling the simulator but also for collecting, summarizing, and analysing the data. A simulation programming language must allow for easy representation and manipulation of simulation time.
>
> (Handling routines to simulate event executions) When a scheduled event is to be executed, the simulation program normally effects the required changes in system state by invoking a program module designed specifically for that purpose. Depending on the complexity of the required changes, the execution could be simulated with one or two statement routines inserted at the appropriate points in the model or with more complex, long routines implemented as sub-programs.
>
> (Managing queues)…A convenient way of representing a queue is by list because the primary operations in queue management are addition and deletion of members. These two operations are easily performed using the list representation and the manipulation of pointers. Thus a language with efficient list-processing capability offers a significant advantage in simulation.
>
> (Collecting data) Many simulation models are implemented to assess the effect on the system of varying certain conditions or parameters. This requirement and comparison requires some facility for the collection of data…Nearly all languages that have gained any degree of recognition have this capability.
>
> (Summarizing and analysing data) In most cases the simple collection of raw data are not enough… Other simulation projects may require a more extensive analysis of the data; perhaps regression analysis, analysis of variance, time series analysis, or other sophisticated tools of statistical analysis. A language that provides this level of analysis as part of its normal operation offers an advantage in simulation. The programmer can spend much time designing and coding the statistical analysis modules if they are not automatically provided in the programming language.
>
> (Formulating and printing output) Some simple simulation models may require only simple output. Others may require extensive report writing…A language that allows flexibility in formatting and representing the data offers a significant advantage, particularly if the model is complex and if the results are to be presented to management personnel.

In addition to these seven common factors, Law et al (Law & Kelton, 1991) also identify an extra factor in selecting a language for simulation -- the assistance provided during the debugging phase of program development. The authors identify that the

special-purpose simulation languages usually provide extensive debugging assistance. As we can see that Graybeal et al discuss only the essential features of the programming language, whereas, Law et al (Law & Kelton, 1991) take into consideration issues germane to the implementation of the programming language too.

Implementation feature in a programming language such as debugging is very important. Emshoff and Sisson (Emshoff & Sisson, 1971) report that it is not uncommon to reduce the time spent on coding and debugging a simulation model by a factor of ten through the use of a language with facilities which enhance the debugging process.

Besides the above factors, we can also identify other factors such as: whether the language is well documented; whether the language is flexible so that the analysts can use it for more than one model; if two languages are both suitable for a simulation projects, then the language with which the programmers are familiar is preferred, because it is impossible for the programmers to write an efficient program with the language they don't know well in a short time.

Graybeal and Pooch (Graybeal & Pooch, 1980) indicate that the relative weights given to these factors depend on the specific problem, and the aim in all cases, is to minimize the time spent in coding and debugging phase.

Taking all these factors into consideration, we have got two choices: to use a general-purpose programming language or a special-purpose simulation language.

5.3.2 General-Purpose Programming Languages

It is quite understandable to choose a general-purpose programming language for implementing a simulation model. One of the main reasons is the availability of the languages, such as FORTRAN, C, and Java. Many computers used in business probably have a compiler or interpreter for these general-purpose languages.

- Java

Java is a high-level, third generation programming language like C, Fortran and many others. We can use Java to write computer applications that crunch numbers, process

words, play games, store data or do any of the thousands of other things computer software can do.

What is most special about Java compared with other general-purpose programming languages is that it lets you write special programs called applets that can be downloaded from the Internet and run within a web browser.

There is a problem with distributing executable programs from web pages. As we know computer programs are very closely tied to the specific hardware and operating system they run. A Windows program will not run on a computer that only runs DOS. Therefore major commercial applications like Microsoft Word or Netscape have to be written almost independently for all the different platforms they run on. Java solves the problem of platform-independence by using byte code. The Java compiler does not produce native executable code for a particular machine

In InterProcs, Java is used as the programming language. The main reason we think is on account of Professor Lee's intention in making the trade scenario as objected-oriented components. Professor Lee wants these components to be combined with other business system components to provide a production level implementation. Then these trade scenarios can be stored in a publicly accessible repository to be downloaded for plug-and-play installation of complex trade transition models, providing companies with easier entry into the global market. For this purpose, Java is a good choice due to its object-oriented characteristics.

- Enterprise JavaBeans (EJB)

In essence, Enterprise JavaBeans (EJB) is a server component model for Java and is a specification for creating server-side, scalable, transactional, and secured enterprise-level applications. Most important, EJBs can be deployed on top of existing transaction processing systems including traditional transaction processing monitors, Web servers, database servers, application servers, and so forth. The EJBs containing the business logic are platform-independent and can be moved to a different, more scalable platform in case of need.

EJB takes a high-level approach for building distributed systems. It helps the enterprise developer concentrate on programming the business logic only. For example, the

developer no longer needs to write code that handles transactional behaviour, security, or threading because the architecture delegates this task to the server vendor.

EJB provides distributed transaction support -- EJB provides transparency for distributed transactions. This means a client can begin a transaction and then invoke methods on EJBs present within two different servers, running on different machines, platforms, or Java virtual machines. Methods in one EJB can call methods in the other EJB with the assurance they will execute in the same transaction context. EJB helps create portable and scalable solutions -- EJBs conforming to the EJB API will install and run in a portable fashion on any EJB server.

As shown in Figure 4.19 in the previous chapter, the trade scenarios designed with InterProcs can be made accessible as libraries on the web. These libraries, along with XML schemas, would be integrated with reusable component architecture such as Enterprise JavaBeans. Further more, this architecture would combine functionality from off-the-shelf and customer developed business applications.

- Prolog

Prolog is a logic language that is particularly suited to programs that involve symbolic or non-numeric computation. For this reason it is a frequently used language in Artificial Intelligence where manipulation of symbols and inference about them is a common task.

Prolog consists of a series of rules and facts. A program is run by presenting some query and seeing if this can be proved against these known rules and facts.

In Prolog we can make some statements by using facts. Facts either consist of a particular item or a relation between items. For example we can represent the fact that it is sunny by writing the program:

Sunny.

We can now ask a query of Prolog by asking:

?- Sunny.

?- is the Prolog prompt. To this query, Prolog will answer yes. Sunny is true because (from above) Prolog matches it in its database of facts.

Prolog is used in the previous version of InterProcs because the audit daemon that is an automated auditing technique contained within InterProcs. The idea is developed by Professor Lee in 1990. The purpose of audit daemon is to inspect a procedure and identify possible control weaknesses and potentials for fraud. According to Professor Lee, these audit daemons have diagnostic function only. How to devise appropriate controls to repair any control weakness is left to the designer of the trade scenario.

Audit daemons consist of deductive rules, which are in logic programming style, plus certain special pattern matching predicts for Documentary Petri Nets. Prolog is chosen for its characteristics on logic programming and symbol manipulating to represent those deductive rules and pattern matching predicts.

So we can identify the advantages of using the general-purpose programming languages. They are: most of the programmers have at least been exposed to these general-purpose programming languages like Java, so the user familiarity is not a problem; models developed in general-purpose programming languages are highly portable because almost all of the high level general-purpose programming languages are platform-independent. A model developed at one place can probably be run at another location without changes or with only minor changes.

Another advantage, which is very hard to quantify for a particular project, is that programmers using one of the general-purpose programming languages to develop a simulation model are likely to be more aware with the details of the model than if the model was coded in one of the special-purpose simulation languages. Since the programmers are required to address all aspects of the model, the programmers will be obviously familiar with the design details.

The main drawback of general purpose programming languages is that the entire model must be coded by the programmers. General-purpose programming languages provide few simulation-oriented functions and little debugging assistance other than pointing out syntactic errors.

5.3.3 Special-Purpose Simulation Language

Graybeal and Pooch (Graybeal & Pooch, 1980) state that the special-purpose simulation languages were developed because many simulation projects need similar functions across various applications. Law and Kelton (Law & Kelton, 1991) also think that it is the commonality of those factors we mentioned above that led to the development of special purpose simulation languages. The authors stated:

> The improvement, standardization, and greater availability of these simulation languages has been one of the major factors in the increased popularity of simulation after 1950s in which the special-purpose simulation languages were started to be developed.

GPSS (General-Purpose Simulation Language) (Gordon, 1975) was originally developed by Geoffrey Gordon at the IBM in 1961. It is well suited for queuing systems. In the 1960s and 1970s, GPSS was a very popular simulation language, this is because the queuing nature of many simulation models at that time, IBM's strong influence on the industry, and GPSS's being taught in many University simulation course.

SIMAN (SIMulation ANalysis) (Pegden, Sadowski, & Shannon, 1990) was developed by Dennis Pegden in 1982. SIMAN gained acceptance quickly because it was the first major simulation language to be available for microcomputers and also because of its special features for manufacturing, including work stations, transporters, conveyors, and automated guided vehicles.

SIMSCRIPT (Law & Larmey, 1984) was developed by Harry Markowitz et al in 1962. It evolved through a number of versions, with the latest one, SIMSCRIPT II.5, being marketed by CACI Products Company. SIMSCRIPT II.5 is a general programming language containing the capabilities for building discrete event, continuous, or combined simulation models.

Each special-purposes simulation language has its own characteristics, and is suitable for a special category of simulation models. For SIMSCRIPT II.5, its English-like and free form syntax make SIMSCRIPT II.5 simulation programs easy to read and almost self-documenting. Because of its sophisticated data structures, and its powerful control statements, SIMSCRIPT II.5 is more suitable for large, complex simulation model than another other special-purpose simulation language.

Using of special-purpose simulation languages has some advantages over the general-purpose programming languages. Simulation languages automatically provide most of the features needed in programming a simulation model, such as advancing simulation time and collecting data, etc. resulting in significant decrease in programming time.

Law and Kelton (Law & Kelton, 1991) state that simulation models are generally easier to change when written in a simulation language; also most of the simulation languages provide dynamic storage allocation during execution. Simulation languages provide a natural framework for simulation modelling; their basic building blocks are more closely similar to simulations than those in a language like Java.

Simulation languages provide better error detection because many potential types of errors have been identified and are checked for automatically. Since fewer lines of code have to be written, the chance of making an error will probably be smaller.

The drawback of special-purpose simulation languages is quite obvious. These simulation languages are not familiar by many people as to general-purpose languages, so the user familiarity is a problem. Training should be given to the programmers before the modelling project. Usually these software packages are costly.

The choice of a suitable programming language is an important consideration in the development of a simulation model. The choice must be made based on the characteristics of the individual project. Considerations such as the languages supported at the installation, the programmer's level of proficiency, and the complexity of the model being developed all have an impact on this decision.

Simulation languages can substantially reduce both programming and project time. By design, they offer language, program, and data structures that make models much easier to develop and modify than that with the general-purpose programming language. That is the reason why we prefer special-purpose simulation language to general-purpose programming language.

Further, our view is that this simulation language should be English-like, self-documenting, and readable by the user for our demonstration purpose. The users, in our case, maybe a manager who uses the model for decision making or someone who is primarily interested in the system under study, but not computer programming.

Users may want to play around with the model for better understanding, such as adding extra documents to see what the system behave, or add more roles such as banks or insurance companies to see what will be interested in the system, etc. That is another reason why we prefer the SIMSCRIPT II.5 that is English-like, self-documenting, and readable programming, to other special-purpose simulation languages.

An aspect of simulation languages is the world-view of the model through this specific language. With a general-purpose language, such as Java, this means to look at the data structures of the language; but with a simulation programming language like SIMSCRIPT II.5, it involves much more.

The world-view of a simulation language, in a sense, defines the class of problems for which the language is suitable. Another way of characterizing the world-view of the language is: how one views the world with the language. For example, in SIMSCRIPTII.5, one begins to describe the modelled system in terms of entities. These entities are characterized by their attributes. If there are logical associations or groupings of entities, they are described as sets. The actions in the modelled system are described as events or processes.

Logical consistency tests are performed, some during compilation of the model and some during execution of the model, which are related to the correct application of the world-view, not just to programming details.

The world-view of the language, together with the support of the constructs by the compiler, and debugger, are the most important reasons for us to choose SIMSCRIPT II.5 as a simulation language for our GT3 model.

In this section, we discussed some common factors for program language being used in simulation; also we gave some advantages of using special-purpose simulation language over the using of general-purpose programming language. Lastly, we chose SIMSCRIPT II.5 as our programming language in GT3 implementation. That is because SIMSCRIPT II.5 automatically provides most of the features needed in programming a simulation model. Also it is English-like, readable by users, etc. In the next section we will discuss the implementation of GT3 in SIMSCRIPT II.5.

5.4 GT3 in SIMSCRIPT II.5

In this section we are going to implement GT3 model in SIMSCRIPT II.5. Before we start, we set up three goals in order to evaluate our implementations. With these goals, we will do evaluation on InterProcs and our implementations of GT3.

5.4.1 Implementation Goals

In the model development phase, for the convenience of the high participation of non-technical people in Business Process Reengineering team, ease of design and modification is very important. It is highly desired that the implementation should be user-friendly so that the Business Process Reengineering specialists can design the model freely without thinking about any constraints in terms of computer programming to avoid time-consuming trail and error. Also a visual display and animation of the behaviour of the model is preferred and the ability to experiment with the model by changing any part of it is highly demanded. So we set up our first goal here as: ease of design and modification.

We set up our second goal as: the correctness of implementation. To study a system through simulation can be efficient and much safer than to experiment directly with the real system. This is based on the assumption that the simulator should reflect the real system in a satisfactory level, that is, what we simulate is quite close to what happens in the real system. In our case we need to determine whether the results obtain from simulation are sufficiently accurate by comparing them with the real world result. This comparison method can be applied to our implementation because our model is a simplified one that can be simulated by hand.

The third goal we set up is: full environment support for running simulation. The implementation should have the ability to collect a variety of statistics to measure the performance of the simulated system. It should also have the ability to generate randomness easily to simulate the variability. Furthermore, users should have full control on the simulation, which means simulation can be run faster, slower or step by step for analysis purpose.

In this section, we set up three goals to evaluate our implementations. The three goals are: ease of design and modification; the correctness of implementation; full

environment support for running simulation. According to these three goals, we first evaluate our baseline implementation -- Professor Lee's InterProcs.

5.4.2 Evaluation of Baseline Implementation -- InterProcs

As mentioned in Chapter 4, we experimented with InterProcs.22feb2001 which is the thirteenth version released in Feb. 2001, on GT3 for several times. But this version is not very stable, so it gives us some difficulties on evaluation. But our thoughts are that with new versions of InterProcs released, maybe this problem can be fixed. So we intend to give a fair evaluation on Professor Lee's work.

We have introduced how to design a new trade scenario step by step in Chapter 4. InterProcs provides a graphic user interface; it is easy for a non-technical people to use it to design a new trade scenario if several hours of training are given. For modification side, we have no way to evaluate it now. When I modified the model, InterProcs.22feb2001 stopped working. At this stage we can only say that InterProcs.22feb2001 achieved our first goal partially, that is, ease of design. But we don't know how it reacts with modification on the model.

For our second goal: the correctness of implementation, we use examples provided by Professor Lee along with the InterProcs.22feb2001 release to hand simulate the system and compare the results with the corresponding simulations running in InterProcs for several times. As we note in trade scenario, interactions among roles are through documents only, transitions are triggered by tokens in state and document places. Results from hand simulation show that all the documents are processed exactly as the same order they should be processed in the real world application. So we think InterProcs.22feb2001 correctly simulates the real world system's behaviours. In this point of view, we can say that InterProcs.22feb2001 achieves our second goal.

InterProcs.22feb2001 does not provide the ability to collect data for statistic purpose to measure the performance of the simulated system. Also users do not have full control on the execution of the simulation. Because when simulating trade scenarios using InterProcs.22feb2001, human interactions are needed to check whether the documents content is right or not. Users must follow the simulation step by step, how fast the simulation can be executed depends on how quick the user can handle the documents.

So InterProcs.22feb2001 does not achieve our third goal. It does not provide full environment support for running simulations.

In this section, we evaluate Professor Lee's implementation – InterProcs. Conclusions are listed below. InterProcs.22feb2001 partially achieves our first goal; ease of designing new trade scenario, but it is not clear for modification of existing trade scenarios. InterProcs.22feb2001 achieves our second goal by correctly describing how the real world system works through simulation. InterProcs.22feb2001 does not achieve our third goal, because it does not provide full environment support for running simulation.

In the next section we briefly introduce the installation of SIMSCRIPT II.5, from then on we will start our SIMSCRIPT II.5 implementation on GT3.

5.4.3 SIMSCRIPT II.5 Release 2.0 for Windows Installation

In order to run SIMSCRIPT II.5 (CACI, 1997b), an 80386 or later processor is needed. Operating System requires Windows 95 or Microsoft Windows NT Version 4.0 or later. A minimum of 100 MB available on hard disk is also recommended to install SIMSCRIPT II.5, a C compiler, sample programs and online documentations. The machine used during these evaluations is a Pentium II with Windows 2000 Professional installed.

To run SIMSCRIPT II.5, the professional version of Microsoft Visual C/C++ Version 5.0 or later is also required. SIMSCRIPT II.5 for Windows Release 2.0 requires the typical configuration of the C compiler. In my machine, Microsoft Visual Studio 6.0 is installed.

Before installing SIMSCRIPT II.5 Release 2.0, remove any older version of SIMSCRIPT II.5 which has been installed previously to avoid conflicts. SIMSCRIPTII.5 will be installed automatically into \SIMSCRIPT.

In the process of installation, we choose typical installation, so all the online documentation, example programs, and tutorial program demonstration programs are all installed. SIMSCRIPT II.5 is available for PC and mainframes. This Release 2.0 for Windows version is embedded in the SimLab package, which is an interactive,

multitasking-programming environment for facilitating the use of SIMSCRIPT II.5. It contains an editor, the SIMSCRIPT II.5 compiler, a debugger, and online help.

When the install is complete, all the SIMSCRIPT II.5 integrated tools including the compiler, the SIMGRAPHICS graphical editor SimDraw, the SIMGRAPHICS graphical editor SimEdit, and Datagraph can be activated through SimLab.

After installation we must modify the environment to add SIMSCRIPT\BIN to the PATH environment variable, SIMSCRIPT\LIB to the LIB variable and SIMSCRIPT\INCLUDE to the INCLUDE environment variable. The PATH, LIB and INCLUDE environment variables also need to be modified to include entries which point to the C++ compiler.

On the machine in use these environment variables are as follows:

Variables	Value
INCLUDE	…;c:\Program Files\Microsoft Visual studio\VC98\include; c:\SIMSCRIPT\include;…
PATH	…;c:\Program Files\Microsoft Visual tudio\VC98\bin; c:\SIMSCRIPT\bin;…
LIB	…;c:\Program Files\Microsoft Visual Studio\VC98\lib; c:\SIMSCRIPT\lib;…

Table 5.1 Environment Variables and their Corresponding Values.

After Installation of SIMSCRIPT II.5 and License Management, we can test the new installation by executing the SimLab package. Also we can run some sample programs shipped with SIMSCRIPT II.5 for Windows release 2.0 to test the new installation.

5.4.4 First Implementation of GT3 in SIMSCRIPT II.5

In SIMSCRIPT II.5 the notion of a process is used as the primary dynamic object. A process represents an object and the sequence of actions it experienced in the model. A process object enters a model at an explicit simulation time when it is created. It becomes active either immediately or at a prescribed activation time.

Processes interact either implicitly or explicitly. When we say implicitly, we mean they interact through resource competition. When we say explicitly, we mean they interact through executing statements to activate, interrupt, or resume one another.

In our model, we map each action of a role as a separate process. That means, in Importer role, we have got three processes called ImporterP1, ImporterP2, and ImporterP3 corresponding to actions Send Purchase_Order, Send Payment, and Receive Payment_Ack respectively. Examples of mapping are illustrated in Figure 5.4. With Exporter, we have two processes call ExporterP1 and ExporterP2 corresponding to Send Goods Delivery and Send Payment_Ack. In Transporter we have only one process called TransporterP1 related to action Send Goods. Also in this GT3 model, we need a generator process to generate new Importer instances automatically.

In this mapping style, ImporterP1 is activated whenever there is a new Importer instance is generated by the Generator process; ExporterP1 process is activated when a purchase order is produced by ImporterP1 process and so forth. These are shown in Figure 5.5.

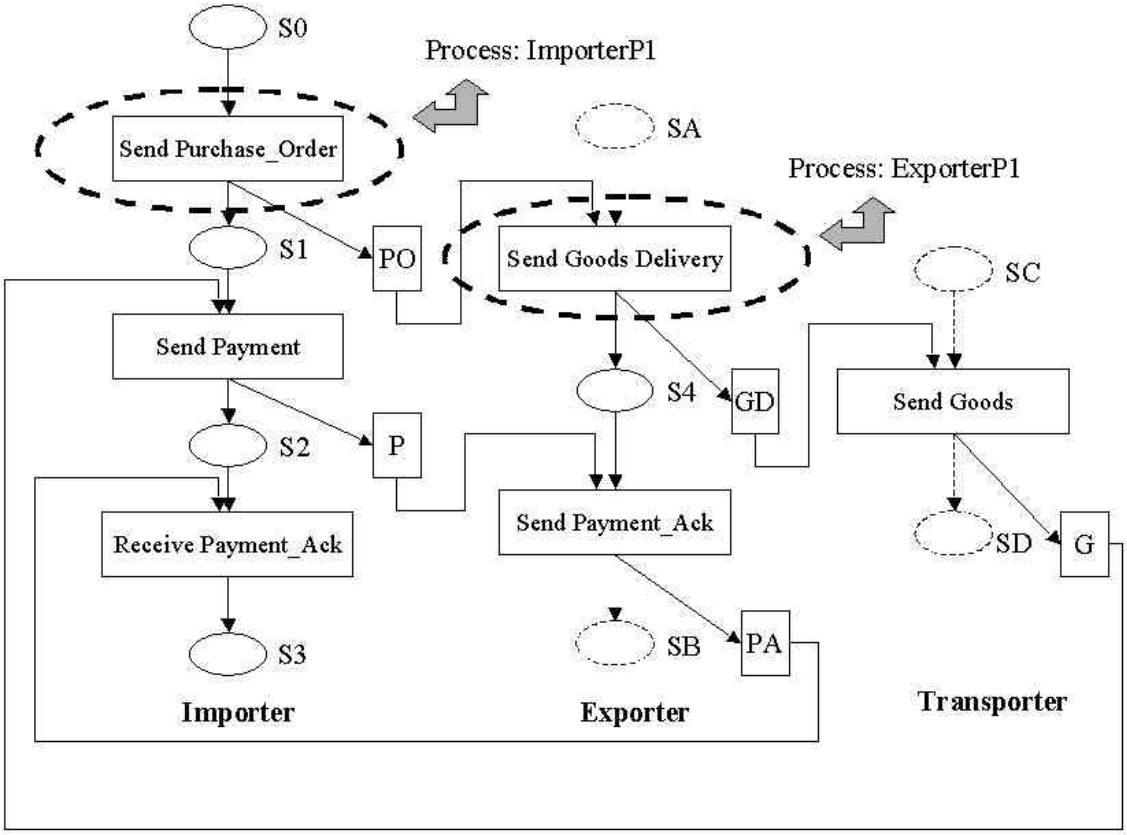


Figure 5.4 Examples of Mapping Each Action as a Process.

At the very beginning, it is quite natural to map all the actions of a role to a big process because these actions are all performed by the same role. So we may map all actions performed by Importer into a single process called ImporterP shown as a vertical dashed circle in Figure 5.6. Therefore we get the other two processes ExporterP and TransporterP. We will show this mapping style results in a very complicated structure in each process.

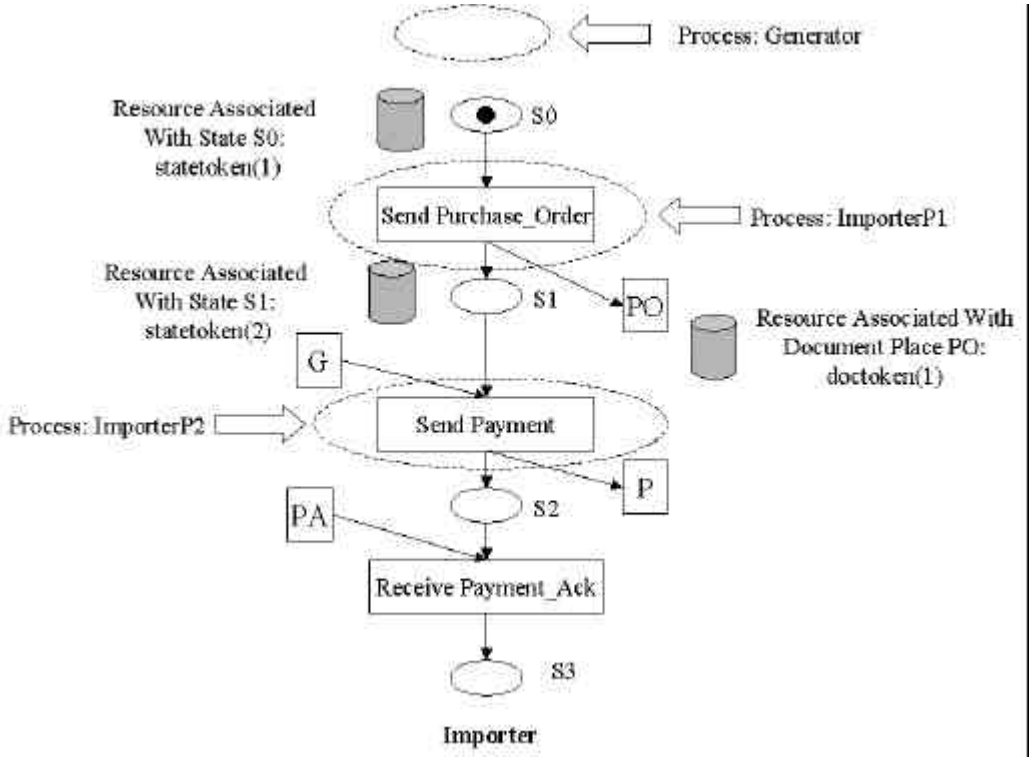


Figure 5.5 Examples of Resources Associated with some State Places.

We walk through a few steps in this model, as an example to get an idea what will happen if we map all the actions Importer performs to a big process. As illustrated in Figure 5.6, when the ImporterP process is activated, it sends the purchase order out. Then it needs to activate ExporterP process and suspend itself waiting for Goods delivered by the TransporterP. As soon as ExporterP process is activated, it will produce a goods delivery instruction and send it to TransporterP process. After that ExporterP process will activate TransporterP process and suspend itself to wait for the payment from ImporterP process. When the TransporterP process is being activated, it sends Goods to ImporterP. Also TransporterP process suspends itself and reactivates ImporterP process, at this stage, the ImporterP process is resumed.

Along this way walking through the whole trade procedure, we can find that there are two suspends and resumes in Importer process and one suspend and resume each in the Exporter process even in our very simple model. When models involving many roles and each role interacts frequently with other roles, this mapping style will result in a very big process routine. This resulting process is error-prone and very hard to debug due to its fairly complicated structure.

One of the advantages we can get from mapping each action of a role as a separate process is that this mapping makes each process simple and easy to be managed. For example, process ImporterP1 only deals with only one activity -- sending a purchase order and then wait for the next Importer instance to come. This also reduces the interaction times between processes. Each process only needs to interact with two processes -- its predecessor process and successor process once each.

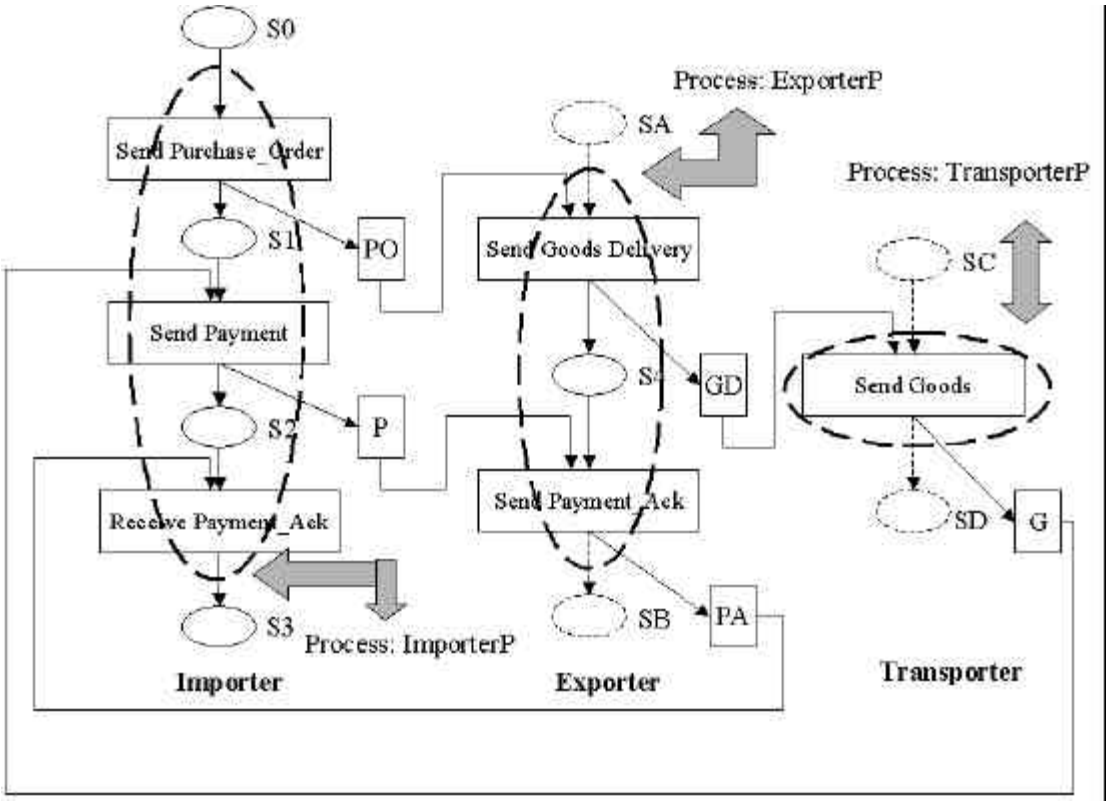


Figure 5.6 Mapping All Actions of Each Role as a Whole Process.

Secondly, mapping each action of a role as a separate process makes each process routine more structural and self-contained. So the simulation program is easy to be debugged and modified. Users can add an action to a role by just copying and doing some modifications on the existing code.

Finally, breaking the whole process into several small processes facilitates parallel computing and makes the model execution more efficient. So mapping each action of a role to a separate process can results in a simple and more structural process routine, and makes it easy to debug and modify the simulation program to suit the user's need.

Resources are the passive elements of a model. A resource is used to model an object, which is required by the processes. Because in our model we map each action of a role as a separate process, communications between these processes are via documents only, so it is quite appropriate for us to describe the interactions between these processes are through resource competition instead of through executing statements to activate, interrupt, or resume one another directly.

If the resource is not available when required, the process object is made to wait until the resource becomes available. A resource becomes available when the process holding it relinquishes it. The process object waiting for the resource is then given the resource and reactivated. If a resource is relinquished when no process object is waiting for it, it is merely made available to be allocated when requested.

Here we give an example to illustrate what kinds of resources are needed in our implementation. In Figure 5.5, the dashed line circles represent some of the processes in our SIMSCRIPT II.5 implementation of GT3. They are: process Generator for generating new Importer instances, process ImporterP1 for producing a purchase order in document place PO and a token in state S1, process ImporterP2 for producing a payment to Exporter in document place P and a token in state place S2.

As an example, we have a close look at process ImporterP1. When Generator process is activated, it produces a new Importer instance represented as a black token and sends out the token in state place S0. In our implementation, this is represented as a black token being placed in the state place S0. Whenever an Importer instance has been generated, it means that there is a black token being put in state S0, at this time, if ImporterP1 process is activated, the black token in state S0 is consumed by ImporterP1, then ImporterP1 can perform its predefined tasks. Here the predefined tasks are to produce a purchase order in document place PO and a black token in state place S2.

Here we can see Generator interacts with ImporterP1 via S0. Without a black token in S0, even if ImporterP1 has been activated, it must be suspended and wait for that black

token be put there by Generator. Interaction between ImporterP1 and ExporterP1 via document place PO is similar. So it is quite natural to map these places as resources.

In Figure 5.5, shadowed cylinders represent some of the resources associated with corresponding state places. statetoken(1) is the name of the resource associated with state place S0, statetoken(2) is the name of the resource associated with state place S1, doctoken(1) is the name of the resource associated with document place labelled as PO.

```
preamble

normally mode is undefined

processes include  Generator, ImporterP1, ImporterP2, ImporterP3,
                   ExporterP1, ExporterP2, TransporterP1

resources include doctoken, statetoken

accumulate AVG.QUEUE.LENGTH as the average
and MAX.QUEUE.LENGTH as the maximum of N.Q.doctoken

temporary entities
                every Document.entity has a status, a name, a receiver, a sender
                define status as an integer variable
                define name, receiver and sender as text variables

define .TIME.UNIT to mean 10
define POINT as a 2-dim real array

display entities include Document.entity, IMAGE1, IMAGE2, IMAGE3, IMAGE4, IMAGE5,
                IMAGE6, IMAGE7, IMAGE8, IMAGE9, IMAGE10, IMAGE11, IMAGE12,
                …..
…
end "preamble
```

Figure 5.7 Part of the Preamble in Our GT3 Implementation.

When the simulation starts, process Generator is executed. Generator will first request this resource -- statetoken(1). At this time, statetoken(1) is free, so process Generator holds it for a predefined time to generate an Importer instance. Then Generator will then release the resource – statetoken(1), at the same time it hands over the Importer instance to process ImporterP1 for further processing. What we can see from the simulation now is a black token is placed in the state place S0.

At this stage, process ImporterP1 can grab the resource associated with the state place S0 -- statetoken(1), take over the Importer instance. Process ImporterP1 can only take over the Importer instance after Generator has generated this instance, this means, until

there is a black token in state place S0 and resource statetoken(1) is available, process ImporterP1 will be delayed indefinitely.

Until so far, we have made our decisions on basic modelling elements: mapping each action of a role as a separate process; mapping each document place and state place as a resource. Now we will have a look at a part of our simulation code for GT3.

SIMSCRIPT II.5 program consists of three primary elements, a preamble that gives a static description of each modelling element; a main program with which the simulation starts; and process routines for which have been declared in the preamble. Figure 5.7 illustrates part of the preamble of our GT3 implementation in SIMSCRIPT II.5.

The first section of this SIMSCRIPT II.5 program is the preamble. It is purely declarative and includes no executable statements. All the modelling elements including processes, and resources being used in our model are declared in this preamble.

We define six processes and two types of resources being used in our SIMSCRIPT II.5 program. Process TransporterP1 maps the “Send Goods” action in Transporter; process ExporterP1 and ExporterP2 map the “Send Goods Delivery” and “Send Payment_Ack” actions in Exporter respectively.

We also define the types of resources in the preamble, doctoken is used to represent the resources associated with document places, and statetoken is used to represent the resources associated with state places. The number of unit of each type will be initialised in main section before the simulation start.

In order to improve readability of programs, SIMSCRIPT II.5 allows us to substitute a character string or a number for a single word. In our preamble, the *define to mean* statement indicates that one .TIME.UNIT equals to ten units of simulation clock time. We can control the simulation speed through increasing or decreasing the number. It is only a convention to have every *define to mean* symbol begin with a period.

The *accumulate* statement provides us with a simple means of specifying which measurements are desired without requiring detailed specification of the method of measurement. An *accumulate* statement is placed in the preamble. Here the *accumulate* statement is used to get the statistics on the queue (average, variance, maximum, etc.) and the utilization of doctoken type of resources.

A *temporary entity* is used to model an object which is short-lived in the model. In our case, documents passing among roles belong to this category. For example, a purchase order is unique for a specific trade procedure. When a purchase order is received by the Exporter correctly, this document becomes meaningless. Display entities defined in our preamble are for animation purpose.

The second part of a SIMSCRIPT II.5 program is main. Our main program is illustrated in Figure 5.8. Execution of a SIMSCIPT II.5 program begins with the first statement in the main program. In INIT routine, we have mainly routines for drawing background, roles, actions, and initialising variables.

In begin routine shown in Figure 5.9, we create and initialise all resources need to be used by processes. Also we give the number of unit of each type of the resources as one by statement: let U.doctoken(I) = 1. Here doctoken type of resource is created and initialised as an example, statetoken type of resource can be done in the same way.

```
main

"---- since there is no quit button: show ctrl-C message on screen
write as "************************************",/
write as "** Close window to end simulation ",/
write as "************************************",/

call INIT

call begin

start simulation

end
```

Figure 5.8 Program Main in GT3 Simulation.

SIMSCRIPT II.5 requires that something be awaiting execution before a simulation commences. This is done by activating processes in routine begin shown in Figure 5.9. The activate statements create six process objects and place them on the pending list with an immediate time of activation using SIMSCRIPT II.5 statement: *activate a* process name *now*.

A simulation begins when control passes to a system-supplied timing routine. This is done by executing the *start simulation* statement in main program shown in Figure 5.8.

Any statements following the *start simulation* statement will not be executed until the simulation has terminated.

The timing routine is at the heart of discrete event simulation. From a programming prospective, it is this timing routine that ties the entire collection of processes together to perform this simulation procedure.

```
routine begin

Define I as an integer variable

create each doctoken(5)
for I = 1 to 5 do
let U.doctoken(I) = 1
loop
…
Let TIMESCALE.V = 10

activate a Generator now
activate a ImporterP1 now
activate a ExporterP1 now
activate a TransporterP1 now
activate a ImporterP2 now
activate a ExporterP2 now
activate a ImporterP3 now

end "begin
```

Figure 5.9 Part of the Begin Routine in Main.

The timing routine is transparent to the programmer. Figure 5.10 illustrates this timing routine. We redraw this figure from (CACI, 1997a). As can be seen from the figure, processes must be on the pending list before entering into the simulation procedure. Termination will occur either because there is no process on the pending list or because some process executes a stop statement.

The third section of our SIMSCRIPT II.5 program is process routines. Each process declared in the program preamble must be further described by a process routine. Each process is described as a separate routine, which begins with a process statement naming the process. Here we use process ImporterP1 shown in Figure 5.11 to illustrate what a process routine looks like in our implementation of GT3.

It is essential that the description of its activities be contained in the process routine. A process routine may be thought of as a sequence of interrelated events separated by lapses of time, either predetermined or indefinite.

Predetermined lapses of time are used to model such phenomena as the service time. Those two wait statements in our example belong to this category; whereas indefinite delays arise because of competition between processes for limited resources. All request statements in our example are potential sources of indefinite delay. In this case processes will automatically be delayed and send back to the pending list until the resource is made available to them.

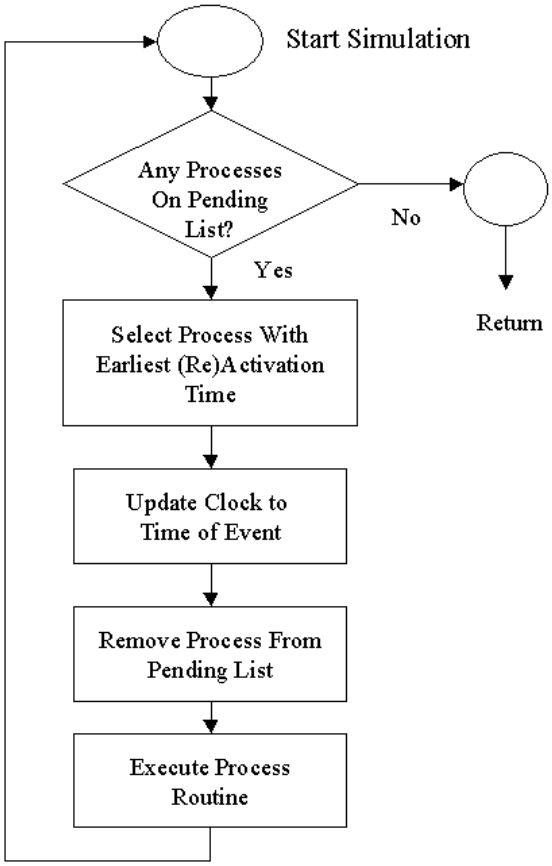


Figure 5.10 Basic SIMSCRIPT II.5 Timing Routine.

The process ImporterP1 describes everything that happens from the time of taking over an Importer instance (consuming the black token in state S0) to the time of producing a purchase order (putting one black token in state place S1, another black token in document place PO). This process routine is very simple in this example because we map each action in a role as a separate process.

Figure 5.12 and Figure 5.13 illustrates the activation and execution of process ImporterP1. When process ImporterP1 is activated, it will firstly requests a statetoken(2) resource. Whenever process ImporterP1 gets the resource it wants, the statement next to this request is executed. If the resource is free, then process ImporterP1 holds it and paints a white token in state place S1, this indicates that this resource now is held by ImporterP1 and not available for process ImporterP2.

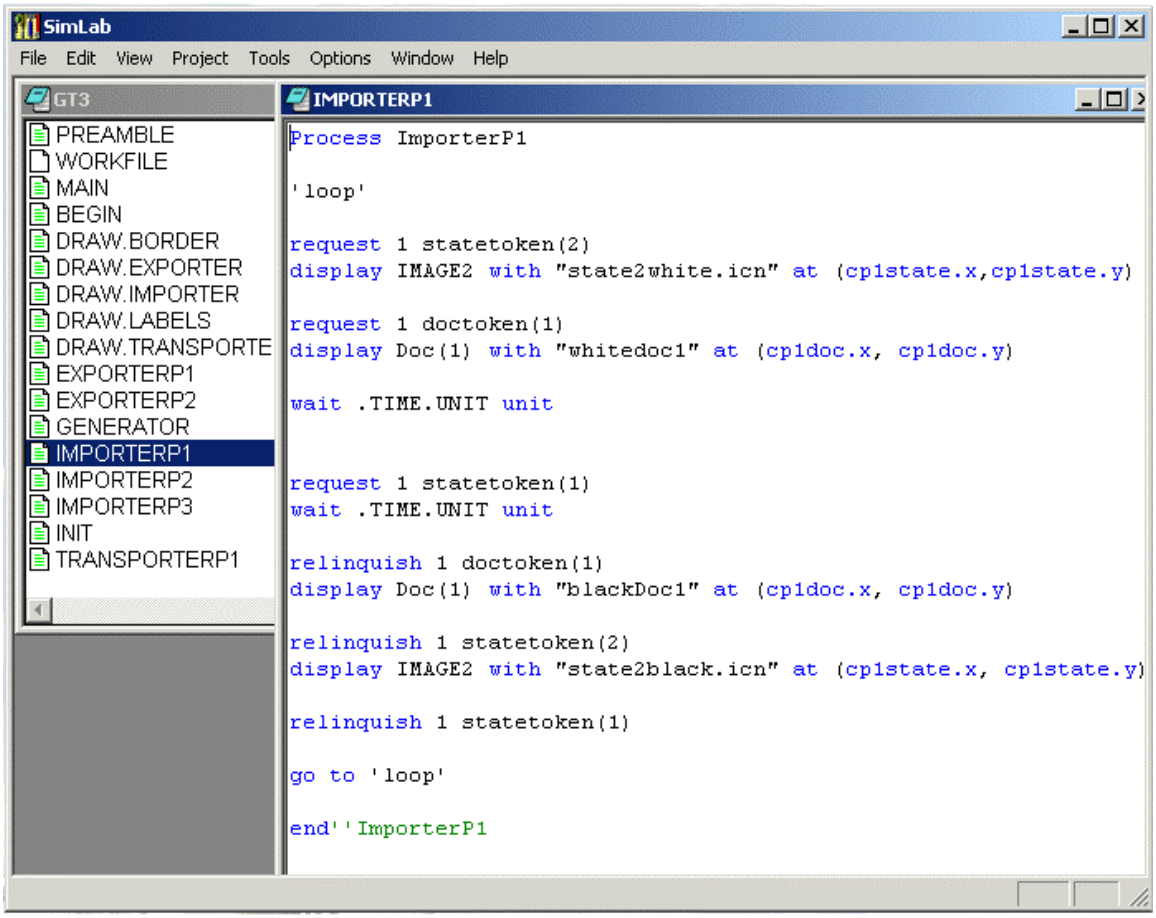


Figure 5.11 ImporterP1: Process Routine Example in GT3 Simulation.

If statetoken(2) is not available, the ImporterP1 process object is automatically placed on a list of objects waiting for statetoken(2) resource. By default this list is ordered as first-come-first-served manner. The process is then suspended until resource statetoken(2) is relinquished by process ImporterP2.

For requesting resource doctoken(1), ImporterP1 works the same way as for statetoken(1). When finally ImporterP1 gets hold of resource statetoken(1), it first suspends itself for one .TIME.UNIT unit to simulate its working time, then it produces

a purchase order representing as putting a black token in document place PO, and relinquishes doctoken(1) to activate process ExporterP1. Process ImporterP1 also produces a black token, puts it in place S1 and relinquish statetoken(2) for ImporterP2. Lastly ImporterP1 relinquishes statetoken(1) to activate Generator.

SIMSCRIPT II.5 also has an interactive symbolic debugger that provides all basic debugging features. We can set break points; use step-by-step execution; and trace, etc. In Figure 5.14 we give the snapshot of GT3 simulation.

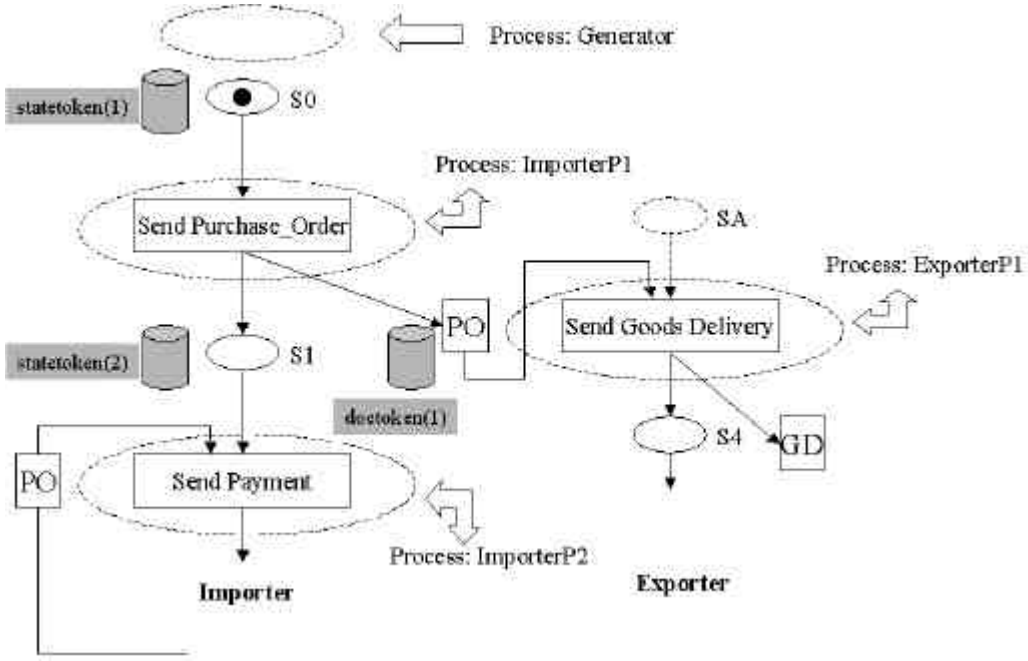


Figure 5.12 Activation of ImporterP1.

Now we will give an evaluation on our first implementation. We start from the first goal: the ease of design and modification. In our first implementation of GT3 in SIMSCRIPT II.5 we can see that, although SIMSCRIPT II.5 is an English-like, self-documenting, and user readable programming language, users still need a lot of training to become familiar with the basic concepts of simulation and the world view of SIMSCRIPTII.5 in order to map the trade scenarios into the basic building blocks of SIMSCRIPT II.5, and program the model into an executable application. SIMSCRIPT II.5 is quite a powerful language for writing a simulator, but the data structure is too complicated for the non-technical users to grasp within a very short time. In this point of view, we evaluate our first implementation in SIMSCRIPT II.5 as not achieved.

For our second goal: the correctness of implementation, we still use the same method as we used to evaluate Professor Lee's work on the second goal. That is, we use the results from the simulator to compare with the results we got from hand simulating of GT3. These hand simulation processes can be illustrated through Figure 4.27 to Figure 4.33 in Chapter 4.

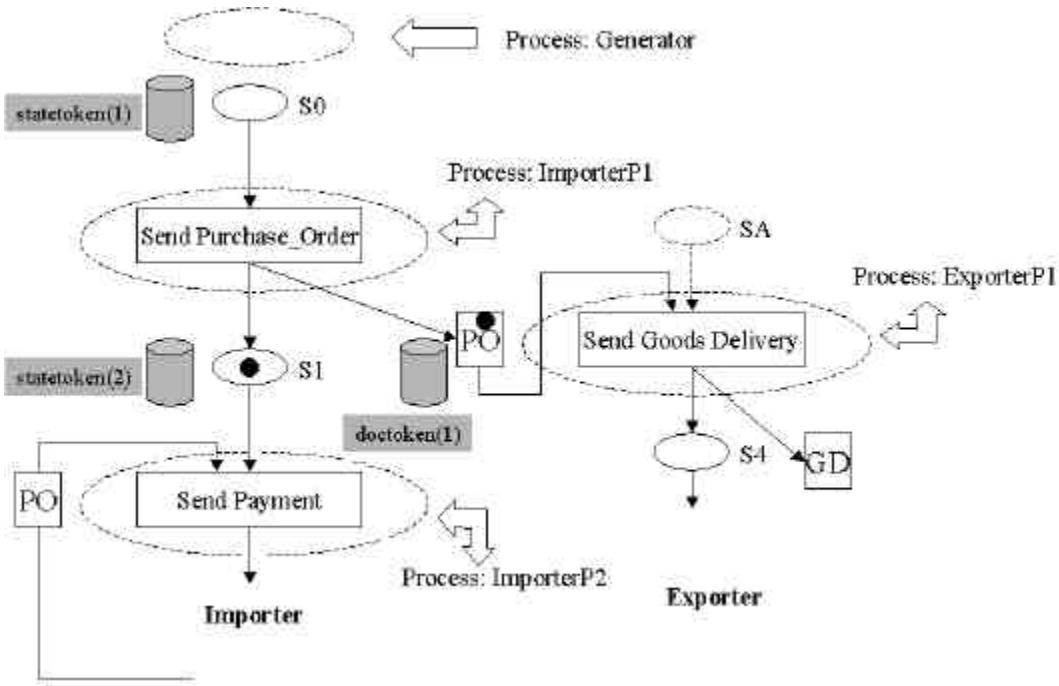


Figure 5.13 Execution of ImporterP1.

When executing simulator in SimLab, which is an integrated developing environment for SIMSCRIPT II.5, a 'Step in' command is provided. This command can help the developer to set up break points for debugging the program. We can use this feature to execute the simulator step by step, then to compare it with the hand simulation results. After several repetition, we found that our first implementation correctly reflect the behaviours of GT3, all documents involved in GT3 are processed exactly as the same order they should be processed in the real world application. So we conclude that our first implementation correctly simulates the real world system's behaviour, our first implementation of GT3 in SIMSCRIPT II.5 achieved our second goal.

For the third goal: full environment support for running simulation. In our first implementation, users have full control over the simulation. Simulation can be run faster, or slower by adjusting a timescale parameter in SIMCRIPT II.5; in our begin routine shown in Figure 5.8, statement: "Let TIMESCALE.V = 10" is for this purpose.

Increase or decrease this integer can make the simulation faster or slower. User can also run the simulation step by step using the 'Step in' command we mentioned above to observe the workflow change at each state through the black tokens flowing around.

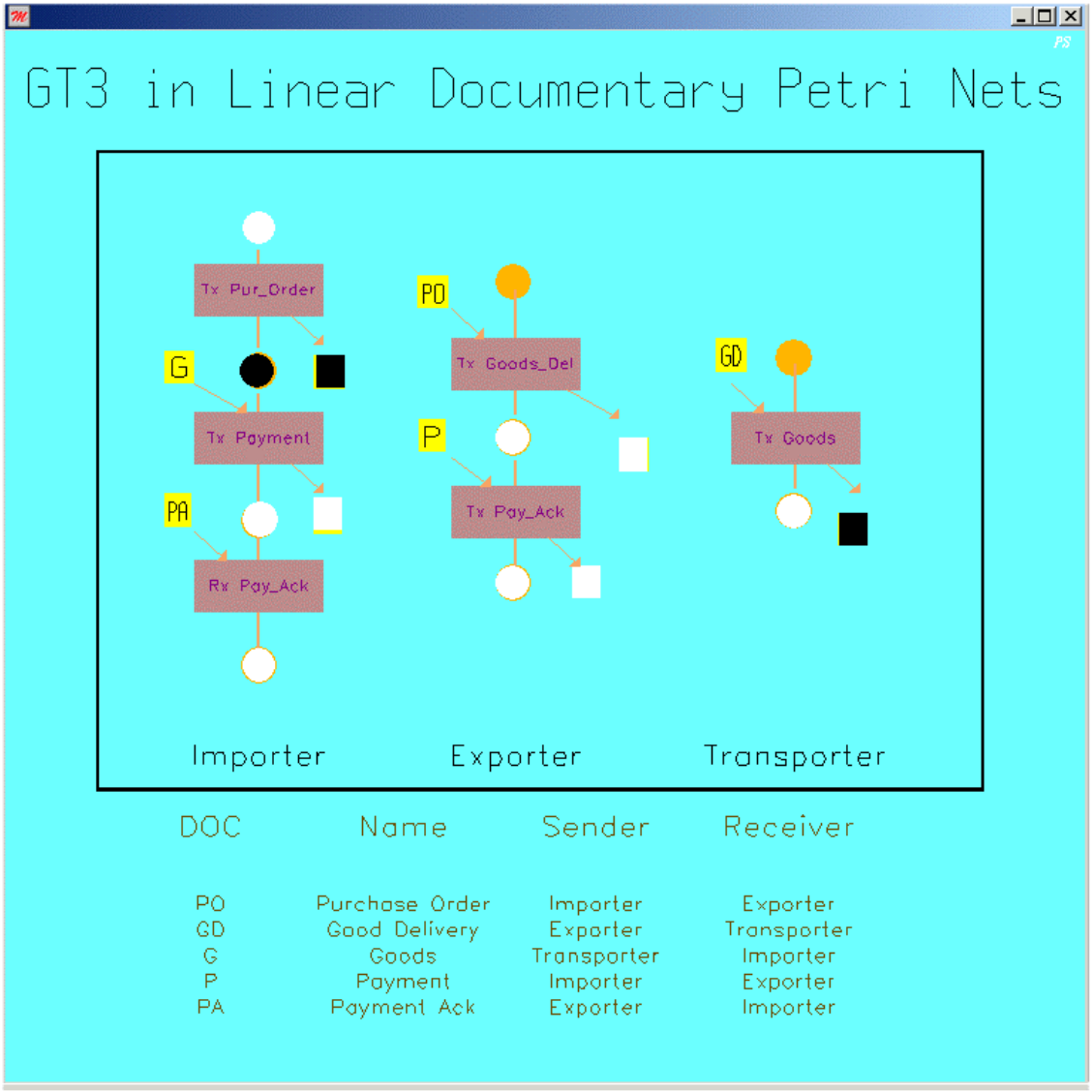


Figure 5.14 Snapshot of GT3 Execution.

Also in our first implementation, we use a uniform time unit "wait .TIME.UNIT unit" shown in Figure 5.11 to simulate the working time for each single process. This working time starts from ImporterP1 being activated, ends at producing a Purchase_Order. In the real world, it is more realistic to postulate that each process has different working times. For example, activity Send_Goods preformed by Transporter may take more time than activity Send Purchase_Order performed by Importer, because

Sends_goods may involve moving, packing and unpacking goods. In this sense, we can use a more suitable way, to generate random working time for each activity.

This can be done easily in our implementation by replacing the statement “wait .TIME.UNIT unit” with random number generating mechanism, such as “wait uniform.f (2.0,8.0,1) minutes”. In SIMSCRIPT II.5 “uniform.f” generates a random number. This function has three parameters: the lower bound, the upper bound, and a random number stream. Each time the function is executed, a new sample from the interval is computed. Random numbers can be used to model the viability in the system. Users can experiment by giving each activity in every process a random working time to see how the workflow changes so as to get better under understanding of the system.

Furthermore, our first implementation also has the ability to collect a variety of statistics to measure the performance of the simulated system. For example, user maybe interested in average or maximum number of Purchase_Orders queuing at resource doctoken(1) for being processed by ExporterP1, this can be done by the “accumulate” statement in preamble shown in Figure 5.7. More statistics measurements can be added according to the user’s interest. At this stage we, evaluate our first implementation achieved the third goal.

In this section, we show our first implementation of GT3 in SIMSCRIPT II.5 through part of our code; also we introduce how to map our scenario to the basic building blocks of SIMSCRIPT II.5 simulation language. Then we evaluate our first implementation as: first goal, not achieved, because it is not easy for users to design and modify the trade scenario; second goal, achieved, the implementation is correct; the third goal, achieved, our implementation provides full environment for supporting running simulator.

Although SIMSCRIPT II.5 is an English-like, and user readable programming language which automatically provides most of the features needed in programming a simulation model, we find it still very hard for general users to use it freely without training. At this stage, we introduce a general-purpose text processing utility -- GEMA, to help general users to experiment with their new ideas on the model and achieve our first goal as well.

In the next section we will discuss our second implementation of GT3. Our second implementation is done with SIMSCRIPT II.5 and GEMA.

5.4.5 Second Implementation of GT3 in SIMSCRIPT II.5 and GEMA

The GEneral purpose MAcro processor (GEMA) is a general-purpose text processing utility based on the concept of pattern matching according to the author of GEMA -- Gray (Gray, 1995). It reads an input file, performs various specified transformations to the data, and copies it to an output file.

Mundie (Mundie, 1996) states that GEMA is a rule-based, non-procedural language. As a macro processor GEMA is more general than other widely used macro processors such as cpp or m4. Because GEMA doesn't impose any particular syntax for what a macro call looks like. It can deal with patterns that span multiple lines and with nested constructs. It can also use multiple sets of rules to be used in different contexts.

For example, we can change a name of the roles involved in GT3, such as change “Importer” to ”Customer”, “Exporter” to “Merchant”. This can be done by the command line shown in Figure 5.15. Here, we assume that GEMA has been installed in c:\gema directory and our source code in the first implementation is saved in the file called GT3old.txt. After executing this command line, a new text file called GT3new.txt is generated.

```
c:\gema>gema -i Importer=Customer;Exporter=Merchant GT3old.txt GT3new.txt
```

Figure 5.15 GEMA Command Line for Making Changes to GT3.

This “-i” option indicates case insensitive mode. All letters in templates will be matched without regard to distinctions of upper case or lower case. This copies file “GT3old.text” to “GT3new.text”, replacing each occurrence of the string “Importer” with the string “Customer” and replacing “Exporter” with “Merchant”. This command line argument consists of two transformation rules separated by “;”.

When this “GT3new.text” is compiled and executed, role names Exporter and Importer shown in Figure 5.14 are replaced by Merchant and Customer. The newly generated snapshot is illustrated in Figure 5.16.

Each rule consists of a template followed by "=" and an action. Here the action is just a literal replacement string, but it is called an action instead of a replacement because GEMA can do much more powerful things.

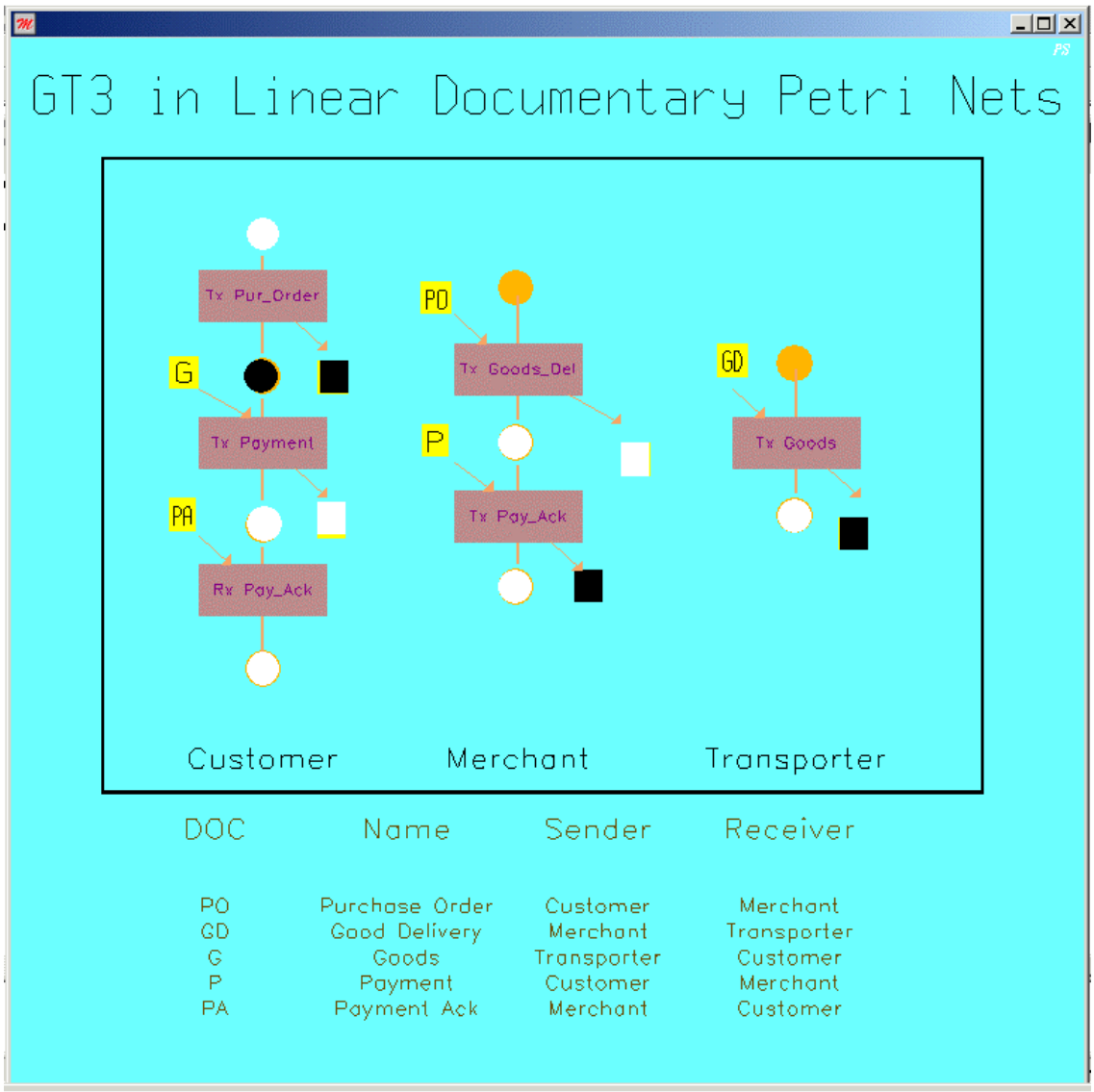


Figure 5.16 GEMA Used to Change the Role Names in GT3.

Command line string replacement is good enough for users to make small changes to the existing models. Our intention here is to allow users to write simple statements according to the predefined structures when they want to design a new trade scenario or to modify an existing trade scenario, then we use GEMA to translate those statements into an executable SIMSCRIPT II.5 program.

In GEMA, patterns can also be defined in one or more files loaded with some kind of option. So we wrote a pattern file for user to translate their structured statements into SIMSCRIPT II.5 program. Because we map each action of a role as a separate process, this makes the process routines in our first implementation very structural. So our second implementation starts from translating these process routines.

Here we use ImporterP1 as an example to show how the translation can be done. The statement shown in Figure 5.17 can be a part of the input file written by user; this simple statement then can be translated into the process routine ImporterP1 in our first implementation, which is shown in Figure 5.11. In the Appendix, we will give the pattern file, but we will not involve too much detail at this stage.

```
pstep (ImporterP1, 2, 1, 1, -1, cp1state.x, cp1state.y,cp1doc.x, cp1doc.y)
```

Figure 5.17 Simple Statement Written by the User.

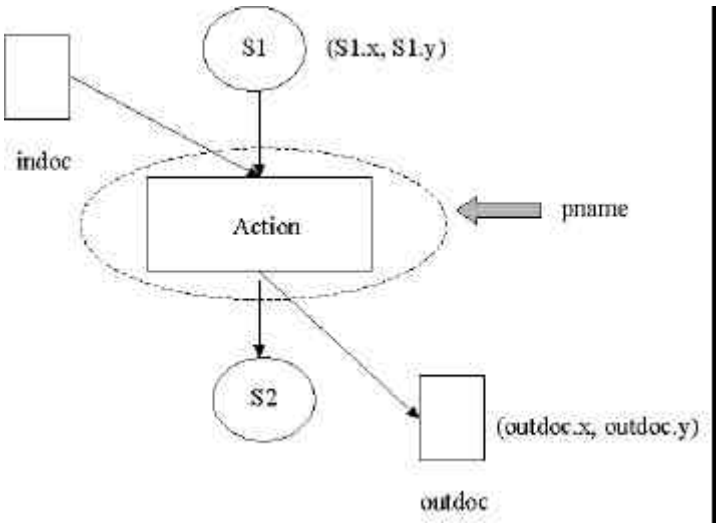


Figure 5.18 Illustration of Predefined Statement Structure.

```
pstep (Pname, S2, S1, outdoc, indoc, S1.x, S1.y,outdoc.x, outdoc.y)
```

Figure 5.19 Statement Structure.

In our second implementation, the emphasis is on achieving the first goal: ease of design and modification. Our intention is to write pattern files in GEMA to translate simple statements, which are written by users for the purpose of designing a new trade

scenario or modifying an existing trade scenario, into executable SIMSCRIPT II.5 program.

We start our work from the translation of process routines. If the user wants to design a new trade scenario, the user firstly needs to depict his trade scenario into a Linear Documentary Petri Net representation. Then user needs to map each action of every role into a separate process. Then each process will look like Figure 5.18.

At this stage, user can write a simple structured statement looking like the one shown in Figure 5.19 to start designing the intended trade scenario. This simple sentence starts with pstep. In the bracket, the parameters are in this order: the first parameter is the name of the process; the second parameter is the sequence number of the outgoing state place. In our case shown in Figure 5.18, the out going state place is S2, the sequence number is 2. The third parameter is the sequence number of the incoming state place. In our case shown in Figure 5.18, the incoming state place is S1, the sequence number is 1. Because we use one type of resource to map the state places, these resources are differentiated by their sequence numbers. The same mechanism is used to differentiate the resources associated with document places.

The fourth parameter is the sequence number of the outgoing document. Usually when a new trade scenario is designed, all the documents in this trade scenario should be assigned a sequence number start with 1. In our GT3 model, PO is the first document to be generated in our scenario, so it is given the sequence number 1. GD is the second document in GT3, so it is given the sequence number 2 accordingly. The fifth parameter is the sequence number of the incoming document. When there is not a document in incoming or outgoing document place, we use a negative number –1 to represent it.

The sixth and seventh parameters are the coordination of the incoming state place, and the eight and ninth parameters are the coordination of the outgoing document. When these are done, this structured statement can be used as part of input file to our pattern file. The output file then is the SIMSCRIPT II.5 source code for this process. We try to compile all the automatically generated process routines, and they work well without problems.

Implementation / Goal	InterProcs (Java)	Our First Implementation (SIMSCRIPT II.5)	Our Second Implementation (SIMSCRIPT II.5 + GEMA)
First goal: Ease of design and modification	Partially Achieved	Not Achieved	Feasible to Achieve
Second Goal: Correctness of implementation	Achieved	Achieved	Achieved
Third Goal: Full environment support for running simulation	Not Achieved	Achieved	Achieved

Table 5.2 Summary of the Evaluations.

In our second implementation, we finished developing the pattern file for process routine. Yet, we leave pattern files for preamble and other routines for future work because the structure of GEMA is very complicated and we have run out of time. The most important thing here is that we have done some practical work to realize our objectives and have got satisfactory results. Through our work on second implementation, we have identified the advantages of combining the two languages: SIMSCRIPT II.5 and GEMA together to get our work more efficient and understandable. It is quite possible that we can develop a powerful tool in assisting process modelling through model construction for demonstration purpose with a little more effort using SIMSCRIPT II.5 and GEMA. With the second implementation, we think it is feasible to achieve our first goal: ease of design and modification. In our second implementation, we have focused only on the first goal as the second and third goals have already been met with in our first implementation itself. The overall evaluation of InterProcs and our two implementations are given in Table 5.2.

In this section, we do our second implementation by programming in SIMSCRIPT II.5 and GEMA. We write pattern files in GEMA to translate the user written structured simple statements into executable SIMSCRIPT II.5 source code automatically. Experiments with our finished part pattern file, we find that it is quite feasible for our second implementation to achieve our first goal: ease of design and modification.

5.5 Summary

In this chapter, based on GT3 model we set up in Chapter 4 and represented it in Linear Documentary Petri Nets, we walk through the designing procedure of GT3 simulation in SIMSCRIPT II.5 and GEMA. We first set up three goals for the evaluation purpose. They are: ease of design and modification, correctness of implementation, and full environment support for running simulation.

We first evaluated Professor Lee's InterProcs, and get a conclusion that InterProcs partially achieves our first goal, achieve our second and not achieve our third goal

In our first implementation of GT3, we start from mapping the model to basic elements in SIMSCRIPT II.5 language, and end at producing an executable GT3 application for demonstrating our GT3 model.

When evaluating our first implementation, we think that it achieves our second and third goals. Our first implementation is correct and it provides full environment support for running simulation. But it does not achieve the first goal, it is not easy to design and modify a trade scenario without knowing SIMSCRIPT II.5 well. That is because of the complicated data structure in SIMSCRIPT II.5.

In order to make it easier to design and modify a trade scenario, we did a second implementation on GT3. This time our intention was to develop an assistance tool to help users in designing and modifying trade scenarios. We intended to write pattern files in GEMA, which is a macro processor based on pattern matching. These pattern files can accept predefined structure statements written by user in an input file, and translate this input file into executable SIMSCRIPT II.5 source code as an output file automatically.

In our second implementation, due to the time limitation and complex data structure of GEMA, we only finished part of the pattern files, that is, the process routine part. We leave the other two parts: preamble and other routines as our future work. Experiment with our finished pattern file shows that it can correctly translate the user written structured simple statements into executable SIMSCRIPT II.5 process routines. Also we identify the advantages of programming in SIMSCRIPT II.5 and GEMA together for better understanding and efficient construction of business process models for

international trade. We evaluate our second implementation as: it provides full environment support for running simulation, it is correct in implementation and it is feasible to ease the design and modification of trade scenarios. In the next chapter we will summarize the thesis and discuss the future work.

Chapter 6 Summary and Future Work

6.1 Summary

During the last several decades, the market structure for international trade has changed greatly. With trade barriers falling, competition intensifies from overseas competitors. It is necessary to deliver high quality products and services to gain and maintain customer loyalty and reduce costs so as to remain competitive. Reengineering business process is considered by a lot of researchers to be a must in order to survive and prosper in today's competitive world.

To improve a business process, we have three models. One of them is called Business Process Reengineering. It brings about rapid change and dramatic improvement to the current business process. Unfortunately, in practice, researchers found that there is a high failure rate in many Business Process Reengineering projects; techniques and technology for modelling business processes have become very important subjects of study to reduce the high failure rate in Business Process Reengineering projects.

We discussed some techniques currently being used in modelling business processes. Each technique has its own characteristics and is suitable for a particular kind of application. Petri nets model of a business process can describe the modelled system precisely and unambiguously due to its formal semantics and powerful analysis methods. The graphical nature can be used to visualize static and dynamic aspects of business process in a natural manner; Petri nets' graphical nature also supports the communication between people involved in a Business Process Reengineering project.

Workflow management technology is used in a number of different contexts and environments because of its ability to reflect the business process rather than to support or automate just discrete tasks. This results in an improved productivity and flexibility needed for Business Process Reengineering. Matured capabilities of Business Process Modelling techniques and the powerful execution capabilities of workflow

Management System can be used in an integrated manner. This mutual benefit exactly fulfils the requirement of modern Business Process Reengineering efforts: to maintain a very tight connection between real world modelling and model execution in order to react fast to changes in the business environment.

International trade has received some academic attention as an application of Business Process Reengineering. Professor Lee (Lee, 1999) identifies mutual-trust related problems in the open electronic commerce environment while conducting business among parties that have no prior trading relationship in international trade. So there is a high probability that international fraud will occur.

In order to assure the interests of the trading parties involved in international trade procedure, Professor Lee introduced a new concept called electronic trade scenario, which is used to govern the activities of the involved parties in international trade. These electronic trade scenarios are specified by a graphical representation language called Documentary Petri Nets. A case tool called InterProcs is also made available by Professor Lee to assist in designing and prototyping trade scenarios.

In Professor Lee's work, he did not give much detail on how to model and analysis a real world business process via the using of Business Process Reengineering techniques. Also we noticed some problems in InterProcs such as instability and lacking of simulation supported features. So we design a simple trade scenario for international trade, and use this solid example to answer the questions not being discussed and supplement the simulation features mot implemented in Professor Lee's work.

We obtain out GT3 model by simplifying the Documentary Credit Procedure in international trade. There are three actors in it, Importer, Exporter, and Transporter. Interactions among these three roles are by means of documents only. In our GT3 model we have five documents. They are: Purchase_order, Goods_Delivery, Goods, Payment, Payment_Ack.

Our preliminary objective of this thesis is to experiment with some of the currently available techniques and technology to make us familiar with the Business Process Reengineering field and answer some of our questions such as: how to represent international trade with a model, how this model can be implemented in a simulation language, etc.

The trading procedure is initiated by Importer sending a Purchase_Order to Exporter. After receiving and processing this Purchase_Order, Exporter sends Goods_Delivery instruction to Transporter. When goods are ready, Transporter sends Goods to Importer. After Importer receives the Goods, it sends Payment to Exporter. Then Exporter sends a Payment_Ack to Importer after it receives Payment.

For better understanding of GT3, we use several Business Process Reengineering techniques to analyse this model from a different point of view. We also modified Professor Lee's Documentary Petri Net by eliminating some constraints and obtaining a simplified representation language called Linear Documentary Petri Nets.

Linear Documentary Petri Nets are used to represent our GT3 model. Linear Documentary Petri Nets inherit all the nice features of classical Petri nets, such as formal semantics, graphic representation, and powerful analysis methods and tools. Besides, Linear Documentary Petri Nets are more structured and easier to manage. For our purpose of demonstration, Linear Documentary Petri Nets are suitable as a representation language for modelling our GT3 model.

Before we started our first implementation, we set up three goals for the purpose of evaluation. They are: ease of design and modification, correctness of implementation, and full environment support for running simulation. We first used these three goals to evaluate Professor Lee's InterProcs, and got a conclusion that InterProcs partially achieved the first two goals but did not achieve the third one.

In our first implementation, while programming GT3 in SIMSCRIPT II.5, we started by mapping our GT3 model, which is represented by our Linear Documentary Petri Nets to basic building blocks in SIMSCRIPT II.5 language, and then produced an executable GT3 application to simulate the workflows in our GT3 model for the purpose of demonstration. Our evaluation on the first implementation shows that our second and third goals are achieved, our first implementation is correct and it provides full environment support for running simulation. Our first goal was not achieved due to the complicated data structure in SIMSCRIPT II.5.

The main objective of the thesis is to develop a software tool to assist in the efforts of Business Process Reengineering for international trade. With the help of this tool, users can design a new trade scenario and make modifications on existing scenarios. In our

second implementation on GT3, we write pattern files in GEMA, a macro processor based on pattern matching. These pattern files can accept predefined structure statements written by user as input, and translate this input file into SIMSCRIPT II.5 source code as an output file automatically. This output file then can be compiled and executed automatically in SimLab, which is the integrated development environment for SIMSCRIPT II.5. In this way, the corresponding workflows within this particular scenario are simulated automatically.

Due to the time limitation and complex structure of GEMA, only part of the pattern files in our second implementation is complicated, that is, the process routine part. Experiments with our finished pattern file show that it can correctly translate the user written structured simple statements into executable SIMSCRIPT II.5 process routines. Also we identified the advantages of programming in SIMSCRIPT II.5 and GEMA together for better understanding and efficient construction of business process models for international trading.

Our evaluation of the second implementation suggests that it achieved our second and third goals and partially achieved the first goal. That is, it provides full environment support for running simulation, it is correct in implementation and it is feasible to ease the design and modification of trade scenarios.

6.2 Future Work

In our second implementation, we only finished process routine part of the pattern file due to the time limitation and the complex structure of GEMA. For a short-term improvement, the pattern files for the other two unfinished parts: preamble and other routines can be written step by step. It is time consuming to write an efficient pattern file for preamble and other routines because of the complicated data structure in GEMA.

For a long-term improvement, validation of business process model is a necessary capability in a training tool for users to understand the Business Process Reengineering projects better, especially when the business process being modelled is complicated. Validation is the process of ensuring that the model is sufficiently accurate for the purpose of study at hand. In Chapter 4 and 5, we did not deal with validating the model. That is because the process of validation is not to be considered as a stage within a

simulation study, but as a process that continues throughout the Business Process Reengineering project according to Lewis et al (Lewis, Brooks, & Robinson, 2001).

Two kinds of validation can be done to a business process model. According to Pidd (Pidd, 1988), White-box validation is to determine whether the constituent parts of the model represent the corresponding real world elements with sufficient accuracy. This is a detailed check of the model. Black-box validation is used to determine whether the overall model represents the real world process with sufficient accuracy, this is an overall check of the model's operation. Black-box validation is often performed by comparing the output from the model with that of the real system to determine whether they are sufficiently similar.

Appendices: Source Codes of GT3 Implementations

Appendix A First Implementation of GT3 in SIMSCRIPT II.5

Appendix B Second Implementation of GT3 in GEMA

Appendix A

First Implementation of GT3 in SIMSCRIPT II.5

```
" Model: First Implementation of GT3 in SIMSCRIPTII.5
" Author: Dong Qiang
" Supervisor: Professor Clark Thomborson
" The University of Auckland, Auckland New Zealand
" Date: Jan 2002
" ============================================================

preamble
Normally mode is undefined
Processes include  generator, ImporterP1, ImporterP2, ImporterP3,
                ExporterP1, ExporterP2, TransporterP1
resources include doctoken, statetoken
accumulate AVG.QUEUE.LENGTH as the average
and MAX.QUEUE.LENGTH as the maximum of N.Q.doctoken
temporary entities
                every Document.entity has a status, a name, a receiver, a sender
                define status as an integer variable
                define name, receiver and sender as text variables
Display entities include Document.entity, IMAGE1, IMAGE2, IMAGE3, IMAGE4,
                IMAGE5, IMAGE6, IMAGE7, IMAGE8, IMAGE9, IMAGE10,
                Doc3in, Doc4out, Doc2out, Doc3out, Doc1in, Doc1out, Doc2in, Doc4in,
                Doc5in, Doc5out, Doc6in, Doc6out
Define MINUTES to mean units
Define Doc as 1-dim pointer array
Define .TIME.UNIT to mean 10
define POINT as a 2-dim real array
define cp1state.x, cp1state.y, cp1doc.x, cp1doc.y, cp2state.x, cp2state.y, cp2doc.x,
                cp2doc.y, cp3state.x, cp3state.y, gstate.x, gstate.y, mp1state.x, mp1state.y,
                mp1doc.x, mp1doc.y, mp2state.x, mp2state.y, mp2doc.x, mp2doc.y,
                tp1state.x, tp1state.y, tp1doc.x, tp1doc.y, cs.x, ms.x, ts.x, cs0.y, cs1.y, cs2.y,
                cs3.y, ms0.y, ms1.y, ms2.y, ts0.y, ts1.y, gdin.x, gdin.y, gout.x, gout.y, poin.x,
                poin.y, oaout.x, oaout.y, gdout.x, gdout.y, poout.x, poout.y, gin.x, gin.y,
                pain.x, pain.y, paout.x, paout.y, pacin.x, pacin.y, pacout.x, pacout.y
               as integer variables
define V.XLO, V.XHI, V.YLO, V.YHI, cstart.x, cstart.y, cdocstart.x, cdocstart.y,
               mdocstart.x, mdocstart.y, tdocstart.x, tdocstart.y, docum.w, docum.h, mstart.x,
               mstart.y, tstart.x, tstart.y, action.h, action.w, solid.line, width.line,
               width.line.link, state.h, type, color, size, DELTA as integer variables
define XLO, XHI, YLO, YHI as double variables
end "preamble
" ============================================================

main
"---- since there is no quit button: show ctrl-C message on screen
write as "**************************************",/
write as "** Close window to end simulation ",/
write as "**************************************",/
call INIT
call begin
start simulation
end
"============================================================

routine begin
Define I, J as an integer variable
Reserve Doc(*) as 6
Create each doctoken(6)
for I = 1 to 6 do
```

```
let U.doctoken(I) = 1
loop
Create each statetoken(9)
for I = 1 to 9 do
let U.statetoken(I) = 1
loop
For J = 1 to 6 do
Create a Document.entity called Doc(J)
loop
Let TIMESCALE.V = 10

activate a generator now
activate a ImporterP1 now
activate a ExporterP1 now
activate a TransporterP1 now
activate a ImporterP2 now
activate a ExporterP2 now
activate a ImporterP3 now

print 4 lines with AVG.QUEUE.LENGTH(1), MAX.QUEUE.LENGTH(1),
and UTILIZATION(1) * 100. / 2 thus
GT3 MODEL WITH 5 DOCTOKEN RESOURCES
AVERAGE DOCTOKEN RESOURCE QUEUE LENGTH IS *.***
MAXIMUM DOCTOKEN RESOURCE QUEUE LENGTH IS *
THE DOCTOKEN(1) RESOURCE WAS BUSY **.** PER CENT OF THE TIME.

end "begin
" ============================================================

process generator
'loop'

"make nextstate white
request 1 statetoken(1)
display IMAGE1 with "state1white.icn" at (gstate.x, gstate.y)
wait .TIME.UNIT unit
"make currentstate black
relinquish 1 statetoken(1)
display IMAGE1 with "state1black.icn" at (gstate.x, gstate.y)

go to 'loop'
end "generator
" ============================================================

process ImporterP1
'loop'

"make outgoingdoc white
request 1 doctoken(1)
display Doc(1) with "whitedoc1.icn" at (cp1doc.x, cp1doc.y)
"make nextstate white
request 1 statetoken(2)
display IMAGE2 with "state2white.icn" at (cp1state.x, cp1state.y)
wait .TIME.UNIT unit
"ask for currentstate
request 1 statetoken(1)
wait .TIME.UNIT unit
"make outgoingdocblack
relinquish 1 doctoken(1)
display Doc(1) with "blackdoc1.icn" at (cp1doc.x, cp1doc.y)
"make nextstate black
relinquish 1 statetoken(2)
display IMAGE2 with "state2black.icn" at (cp1state.x, cp1state.y)
"release currentstate
relinquish 1 statetoken(1)

go to 'loop'
```

```
end "ImporterP1
" ============================================================


process ImporterP2
'loop'

"make outgoingdoc white
request 1 doctoken(5)
display Doc(5) with "whitedoc5.icn" at (cp2doc.x, cp2doc.y)
"make nextstate white
request 1 statetoken(3)
display IMAGE3 with "state3white.icn" at (cp2state.x, cp2state.y)
wait .TIME.UNIT unit
"ask for incomingdoc
request 1 doctoken(4)
"ask for currentstate
request 1 statetoken(2)
wait .TIME.UNIT unit
"make outgoingdoc black
relinquish 1 doctoken(5)
display Doc(5) with "blackdoc5.icn" at (cp2doc.x, cp2doc.y)
"make make nextstate bla ck
relinquish 1 statetoken(3)
display IMAGE3 with "state3black.icn" at (cp2state.x, cp2state.y)
"release incomingdoc
relinquish 1 doctoken(4)
"release currentstate
relinquish 1 statetoken(2)

go to 'loop'
end  "ImporterP2

" ============================================================


process ImporterP3
'loop'

"make nextstate white
request 1 statetoken(4)
display IMAGE4 with "state4white.icn" at (cp3state.x, cp3state.y)
wait .TIME.UNIT unit
"ask for incomingdoc
request 1 doctoken(6)
"ask for currentstate
request 1 statetoken(3)
wait .TIME.UNIT unit
"make nextstate black
relinquish 1 statetoken(4)
display IMAGE4 with "state4black.icn" at (cp3state.x, cp3state.y)
"release incomingdoc
relinquish 1 doctoken(6)
"release currentstate
relinquish 1 statetoken(3)

go to 'loop'
end "ImporterP3

" ============================================================


process ExporterP1
'loop'

"make nextstate white
```

```
request 1 statetoken(6)
display IMAGE6 with "state6white.icn" at (mp1state.x, mp1state.y)
"make outgoingdoc white
request 1 doctoken(3)
display Doc(3) with "whitedoc3.icn" at (mp1doc.x, mp1doc.y)
wait .TIME.UNIT unit
"ask for incomingdoc
request 1 doctoken(1)
"ask for currentstate
request 1 statetoken(5)
wait .TIME.UNIT unit
"make nextstate black
relinquish 1 statetoken(6)
display IMAGE6 with "state6black.icn" at (mp1state.x, mp1state.y)
"make outgoingdoc black
relinquish 1 doctoken(3)
display Doc(3) with "blackdoc3.icn" at (mp1doc.x, mp1doc.y)
"release incomingdoc
relinquish 1 doctoken(1)
"release currentstate
relinquish 1 statetoken(5)

go to 'loop'
end "ExporterP1

" ============================================================

process ExporterP2
'loop'

"make nextstate white
request 1 statetoken(7)
display IMAGE7 with "state7white.icn" at (mp2state.x, mp2state.y)
"make outgoingdoc white
request 1 doctoken(6)
display Doc(6) with "whitedoc6.icn" at (mp2doc.x, mp2doc.y)
wait .TIME.UNIT unit
"ask for incomingdoc
request 1 doctoken(5)
"ask for currentstate
request 1 statetoken(6)
wait .TIME.UNIT unit
"make outgoingdoc black
relinquish 1 doctoken(6)
display Doc(6) with "blackdoc6.icn" at (mp2doc.x, mp2doc.y)
"make nextstate black
relinquish 1 statetoken(7)
display IMAGE7 with "state7black.icn" at (mp2state.x, mp2state.y)
"release for incomingdoc
relinquish 1 doctoken(5)
"release for currentstate
relinquish 1 statetoken(6)

go to 'loop'
end "ExporterP2

" ============================================================


process TransporterP1
'loop'

"make nextstate white
request 1 statetoken(9)
display IMAGE9 with "state9white.icn" at (tp1state.x, tp1state.y)
"make outgoingdoc white
request 1 doctoken(4)
```

```
display Doc(4) with "whitedoc4.icn" at (tp1doc.x, tp1doc.y)
wait .TIME.UNIT unit
"ask for incomingdoc
request 1 doctoken(3)
"ask for currentstate
request 1 statetoken(8)
wait .TIME.UNIT unit
"make nextstate black
relinquish 1 statetoken(9)
display IMAGE9 with "state9black.icn" at (tp1state.x, tp1state.y)
"make outgoingdoc black
relinquish 1 doctoken(4)
display Doc(4) with "blackdoc4.icn" at (tp1doc.x, tp1doc.y)
"release incomingdoc
relinquish 1 doctoken(3)
"release currentstate
relinquish 1 statetoken(8)

go to 'loop'
end "TransporterP1

" ============================================================


routine DRAW.BORDER
define BORDER as an integer variable
let BORDER = 50
call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
let POINT(1, 1) = V.XLO - BORDER
let POINT(2, 1) = V.YLO - BORDER
let POINT(1, 2) = V.XLO - BORDER
let POINT(2, 2) = V.YHI + BORDER
let POINT(1, 3) = V.XLO
let POINT(2, 3) = V.YHI + BORDER
let POINT(1, 4) = V.XLO
let POINT(2, 4) = V.YLO - BORDER
call FILLCOLOR.R(1)
call FILLSTYLE.R(1)
call FILLAREA.R(4, POINT(*, *))
call CLOSE.SEG.R

call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
let POINT(1, 1) = V.XLO - BORDER
let POINT(2, 1) = V.YHI
let POINT(1, 2) = V.XLO - BORDER
let POINT(2, 2) = V.YHI + BORDER
let POINT(1, 3) = V.XHI + BORDER
let POINT(2, 3) = V.YHI + BORDER
let POINT(1, 4) = V.XHI + BORDER
let POINT(2, 4) = V.YHI
call FILLCOLOR.R(1)
call FILLSTYLE.R(1)
call FILLAREA.R(4, POINT(*, *))
call CLOSE.SEG.R

call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
let POINT(1, 1) = V.XHI
let POINT(2, 1) = V.YHI + BORDER
let POINT(1, 2) = V.XHI + BORDER
let POINT(2, 2) = V.YHI + BORDER
let POINT(1, 3) = V.XHI + BORDER
let POINT(2, 3) = V.YLO - BORDER
let POINT(1, 4) = V.XHI
let POINT(2, 4) = V.YLO - BORDER
```

```
call FILLCOLOR.R(1)
call FILLSTYLE.R(1)
call FILLAREA.R(4, POINT(*, *))
call CLOSE.SEG.R

call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
let POINT(1, 1) = V.XLO - BORDER
let POINT(2, 1) = V.YLO - BORDER
let POINT(1, 2) = V.XLO - BORDER
let POINT(2, 2) = V.YLO
let POINT(1, 3) = V.XHI + BORDER
let POINT(2, 3) = V.YLO
let POINT(1, 4) = V.XHI + BORDER
let POINT(2, 4) = V.YLO - BORDER
call FILLCOLOR.R(1)
call FILLSTYLE.R(1)
call FILLAREA.R(4, POINT(*, *))
call CLOSE.SEG.R

end "DRAW.BORDER

" ============================================================

routine DRAW.LABELS

call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
call TEXTALIGN.R(1, 1)
call TEXTCOLOR.R(1)
call TEXTSIZE.R(1500)
call WGTEXT.R("DOCUMENTARY PETRI NET SIMULATION", 16384,31000)
call CLOSE.SEG.R

call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
call TEXTALIGN.R(1, 1)
call TEXTCOLOR.R(1)
call TEXTSIZE.R(1000)
CALL TEXTFONT.R(1)
call WGTEXT.R("Importer", 7500, 10200)
call CLOSE.SEG.R

call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
call TEXTALIGN.R(1, 1)
call TEXTCOLOR.R(1)
call TEXTSIZE.R(1000)
CALL TEXTFONT.R(1)
call WGTEXT.R("Exporter", 15500, 10200)
call CLOSE.SEG.R

call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
call TEXTALIGN.R(1, 1)
call TEXTCOLOR.R(1)
call TEXTSIZE.R(1000)
CALL TEXTFONT.R(1)
call WGTEXT.R("Transporter", 24200,10200)
call CLOSE.SEG.R

call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
call TEXTALIGN.R(1, 1)
call TEXTFONT.R(1)
call TEXTCOLOR.R(10)
call TEXTSIZE.R(1000)
```

```
call WGTEXT.R("DOC", 6000,8000)
call WGTEXT.R("Name", 12000,8000)
call WGTEXT.R("Sender", 18000,8000)
call WGTEXT.R("Receiver", 24000,8000)
call CLOSE.SEG.R


call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
call TEXTFONT.R(1)
call TEXTALIGN.R(1, 1)
call TEXTCOLOR.R(10)
call TEXTSIZE.R(700)

call WGTEXT.R("PO", 6000,5600)
call WGTEXT.R("Purchase Order", 12000,5600)
call WGTEXT.R("Importer", 18000,5600)
call WGTEXT.R("Exporter", 24000,5600)

call WGTEXT.R("GD", 6000,4800)
call WGTEXT.R("Good Delivery", 12000,4800)
call WGTEXT.R("Exporter", 18000,4800)
call WGTEXT.R("Transporter", 24000,4800)

call WGTEXT.R("G", 6000,4000)
call WGTEXT.R("Goods", 12000,4000)
call WGTEXT.R("Transporter", 18000,4000)
call WGTEXT.R("Importer", 24000,4000)

call WGTEXT.R("P", 6000,3200)
call WGTEXT.R("Payment", 12000,3200)
call WGTEXT.R("Importer", 18000,3200)
call WGTEXT.R("Exporter", 24000,3200)

call WGTEXT.R("PA", 6000,2400)
call WGTEXT.R("Payment Ack", 12000,2400)
call WGTEXT.R("Exporter", 18000,2400)
call WGTEXT.R("Importer", 24000,2400)

call CLOSE.SEG.R
end " DRAW.LABELS

" ========================================================


routine DRAW.IMPORTER
define I as an integer variable

"Draw the Vertical Line
call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 0)
let POINT(1, 1) = cstart.x
let POINT(2, 1) = cstart.y - state.h
let POINT(1, 2) = cstart.x
let POINT(2, 2) = cstart.y - 3 * 4600 "+ action.h
call LINECOLOR.R (2)
call LINESTYLE.R(solid.line)
CALL LINEWIDTH.R (width.line)
call POLYLINE.R (2, POINT(*, *))
call CLOSE.SEG.R

"Draw the Action Boxes
for I = 0 to 2 do

call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
```

```
let POINT(1, 1) = cstart.x - action.w
let POINT(2, 1) = cstart.y - 2300 - I * 4600 + action.h
let POINT(1, 2) = cstart.x + action.w
let POINT(2, 2) = cstart.y - 2300 - I * 4600 + action.h
let POINT(1, 3) = cstart.x + action.w
let POINT(2, 3) = cstart.y - 2300 - I * 4600 - action.h
let POINT(1, 4) = cstart.x - action.w
let POINT(2, 4) = cstart.y - 2300 - I * 4600 - action.h
call FILLCOLOR.R(4)
call FILLSTYLE.R(1)
call FILLAREA.R(4, POINT(*, *))
call CLOSE.SEG.R
loop

"Draw all the Labels
call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
call TEXTALIGN.R(1, 1)
call TEXTCOLOR.R(14)
call TEXTSIZE.R(550)
call WGTEXT.R("Tx Pur_Order", cstart.x, cstart.y - 2300 - 4600 * 0)
call CLOSE.SEG.R

call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
call TEXTALIGN.R(1, 1)
call TEXTCOLOR.R(14)
call TEXTSIZE.R(550)
call WGTEXT.R("Tx Payment", cstart.x, cstart.y - 2300 - 4600 * 1)
call CLOSE.SEG.R

call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
call TEXTALIGN.R(1, 1)
call TEXTCOLOR.R(14)
call TEXTSIZE.R(550)
call WGTEXT.R("Rx Pay_Ack", cstart.x, cstart.y - 2300 - 4600 * 2)
call CLOSE.SEG.R

"draw the arrowed link between the document and the action
call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
let POINT(1, 1) = cstart.x + action.w * 0.5
let POINT(2, 1) = cstart.y - 2300 - action.h
let POINT(1, 2) = cdocstart.x + action.w
let POINT(2, 2) = cdocstart.y + docum.h
call LINECOLOR.R (color)
call LINESTYLE.R(solid.line)
CALL LINEWIDTH.R (width.line.link)
call POLYLINE.R (2, POINT(*, *))
call CLOSE.SEG.R

"Draw the arrow
call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
let POINT(1, 1) = cdocstart.x + action.w
let POINT(2, 1) = cdocstart.y + docum.h
let POINT(1, 2) = cdocstart.x + action.w - size
let POINT(2, 2) = cdocstart.y + docum.h
let POINT(1, 3) = cdocstart.x + action.w
let POINT(2, 3) = cdocstart.y + docum.h + size
CALL FILLCOLOR.R(color)
CALL FILLSTYLE.R(1)
CALL FILLAREA.R (3, point(*, *))
call CLOSE.SEG.R
"end of right up document (purchase)
```

```
"draw the arrowed link between the documment and the action
call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
let POINT(1, 1) = cdocstart.x - action.w
let POINT(2, 1) = cdocstart.y - docum.h
let POINT(1, 2) = cstart.x - action.w * 0.2
let POINT(2, 2) = cstart.y - 4600 - 2300 + action.h
call LINECOLOR.R (color)
call LINESTYLE.R(solid.line)
CALL LINEWIDTH.R (width.line.link)
call POLYLINE.R (2, POINT(*, *))
call CLOSE.SEG.R

"Draw the arrow
call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
let POINT(1, 1) = cstart.x - action.w * 0.2
let POINT(2, 1) = cstart.y - 4600 - 2300 + action.h
let POINT(1, 2) = cstart.x - action.w * 0.2 - size
let POINT(2, 2) = cstart.y - 4600 - 2300 + action.h
let POINT(1, 3) = cstart.x - action.w * 0.2
let POINT(2, 3) = cstart.y - 4600 - 2300 + action.h + size
CALL FILLCOLOR.R(color)
CALL FILLSTYLE.R(1)
CALL FILLAREA.R (3, point(*, *))
call CLOSE.SEG.R
"end of left middle (purchase order ack)

"draw the arrowed link between the document and the action
call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
let POINT(1, 1) = cstart.x + action.w * 0.5
let POINT(2, 1) = cstart.y - 2300 - 4600 - action.h
let POINT(1, 2) = cdocstart.x + action.w
let POINT(2, 2) = cdocstart.y - 4600 + docum.h
call LINECOLOR.R (color)
call LINESTYLE.R(solid.line)
CALL LINEWIDTH.R (width.line.link)
call POLYLINE.R (2, POINT(*, *))
call CLOSE.SEG.R

"Draw the arrow
call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
let POINT(1, 1) = cdocstart.x + action.w
let POINT(2, 1) = cdocstart.y - 4600 + docum.h
let POINT(1, 2) = cdocstart.x + action.w - size
let POINT(2, 2) = cdocstart.y - 4600 + docum.h
let POINT(1, 3) = cdocstart.x + action.w
let POINT(2, 3) = cdocstart.y - 4600 + docum.h + size
CALL FILLCOLOR.R(color)
CALL FILLSTYLE.R(1)
CALL FILLAREA.R (3, point(*, *))
call CLOSE.SEG.R
"end of right bottom (Payment)

"draw the arrowed link between the document and the action
call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
let POINT(1, 1) = cdocstart.x - action.w
let POINT(2, 1) = cdocstart.y - 4600 - docum.h
let POINT(1, 2) = cstart.x - action.w * 0.5
let POINT(2, 2) = cstart.y - 4600 * 2 - 2300 + action.h
call LINECOLOR.R (color)
call LINESTYLE.R(solid.line)
CALL LINEWIDTH.R (width.line.link)
call POLYLINE.R (2, POINT(*, *))
```

```
call CLOSE.SEG.R

"Draw the arrow
call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
let POINT(1, 1) = cstart.x - action.w * 0.5
let POINT(2, 1) = cstart.y - 4600 * 2 - 2300 + action.h
let POINT(1, 2) = cstart.x - action.w * 0.5 - size
let POINT(2, 2) = cstart.y - 4600 * 2 - 2300 + action.h
let POINT(1, 3) = cstart.x - action.w * 0.5
let POINT(2, 3) = cstart.y - 4600 * 2 - 2300 + action.h + size
CALL FILLCOLOR.R(color)
CALL FILLSTYLE.R(1)
CALL FILLAREA.R (3, point(*, *))
call CLOSE.SEG.R
"end of left bottom (payment ack)

end "DRAW.IMPORTER

" ============================================================


routine DRAW.EXPORTER
define I as an integer variable

call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 0)
let POINT(1, 1) = mstart.x
let POINT(2, 1) = mstart.y
let POINT(1, 2) = mstart.x
let POINT(2, 2) = mstart.y - 2 * 4600 "+ action.h
call LINECOLOR.R (2)
call LINESTYLE.R(solid.line)
CALL LINEWIDTH.R (width.line)
call POLYLINE.R (2, POINT(*, *))
call CLOSE.SEG.R

for I = 0 to 1 do

call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
let POINT(1, 1) = mstart.x - action.w
let POINT(2, 1) = mstart.y - 2300 - I * 4600 + action.h
let POINT(1, 2) = mstart.x + action.w
let POINT(2, 2) = mstart.y - 2300 - I * 4600 + action.h
let POINT(1, 3) = mstart.x + action.w
let POINT(2, 3) = mstart.y - 2300 - I * 4600 - action.h
let POINT(1, 4) = mstart.x - action.w
let POINT(2, 4) = mstart.y - 2300 - I * 4600 - action.h
call FILLCOLOR.R(4)
call FILLSTYLE.R(1)
call FILLAREA.R(4, POINT(*, *))
call CLOSE.SEG.R
loop

call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
call TEXTALIGN.R(1, 1)
call TEXTCOLOR.R(14)
call TEXTSIZE.R(550)
call WGTEXT.R("Rx Pur_Order", mstart.x, mstart.y - 2300 - 4600 * 0)
call CLOSE.SEG.R

call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
call TEXTALIGN.R(1, 1)
call TEXTCOLOR.R(14)
```

```
call TEXTSIZE.R(550)
call WGTEXT.R("Rx Payment", mstart.x, mstart.y - 2300 - 4600 * 1)
call CLOSE.SEG.R

"Draw documents
"draw the arrowed link between the document and the action
call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
let POINT(1, 1) = mdocstart.x - action.w
let POINT(2, 1) = mdocstart.y - docum.h
let POINT(1, 2) = mstart.x - action.w * 0.5
let POINT(2, 2) = mstart.y - 2300 + action.h
call LINECOLOR.R (color)
call LINESTYLE.R(solid.line)
CALL LINEWIDTH.R (width.line.link)
call POLYLINE.R (2, POINT(*, *))
call CLOSE.SEG.R

"Draw the arrow
call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
let POINT(1, 1) = mstart.x - action.w * 0.5
let POINT(2, 1) = mstart.y - 2300 + action.h
let POINT(1, 2) = mstart.x - action.w * 0.5 - size
let POINT(2, 2) = mstart.y - 2300 + action.h
let POINT(1, 3) = mstart.x - action.w * 0.5
let POINT(2, 3) = mstart.y - 2300 + action.h + size
CALL FILLCOLOR.R(color)
CALL FILLSTYLE.R(1)
CALL FILLAREA.R (3, point(*, *))
call CLOSE.SEG.R
"end of left up (purchase order)

"draw the arrowed link between the document and the action
call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
let POINT(1, 1) = mdocstart.x - action.w
let POINT(2, 1) = mdocstart.y - 4600 - docum.h
let POINT(1, 2) = mstart.x - action.w * 0.4
let POINT(2, 2) = mstart.y - 2300 - 4600 + action.h
call LINECOLOR.R (color)
call LINESTYLE.R(solid.line)
CALL LINEWIDTH.R (width.line.link)
call POLYLINE.R (2, POINT(*, *))
call CLOSE.SEG.R

"Draw the arrow
call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
let POINT(1, 1) = mstart.x - action.w * 0.4
let POINT(2, 1) = mstart.y - 2300 - 4600 + action.h
let POINT(1, 2) = mstart.x - action.w * 0.4 - size
let POINT(2, 2) = mstart.y - 2300 - 4600 + action.h
let POINT(1, 3) = mstart.x - action.w * 0.4
let POINT(2, 3) = mstart.y - 2300 - 4600 + action.h + size
CALL FILLCOLOR.R(color)
CALL FILLSTYLE.R(1)
CALL FILLAREA.R (3, point(*, *))
call CLOSE.SEG.R
"end of left bottom (payment)

"draw the arrowed link between the document and the action
call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
let POINT(1, 1) = mstart.x + action.w * 0.8
let POINT(2, 1) = mstart.y - 2300 - action.h
let POINT(1, 2) = mdocstart.x + action.w + 1200
```

```
let POINT(2, 2) = mdocstart.y - 4600 + docum.h
call LINECOLOR.R (color)
call LINESTYLE.R(solid.line)
CALL LINEWIDTH.R (width.line.link)
call POLYLINE.R (2, POINT(*, *))
call CLOSE.SEG.R

"Draw the arrow
call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
let POINT(1, 1) = mdocstart.x + 1200 + action.w
let POINT(2, 1) = mdocstart.y - 4600 + docum.h
let POINT(1, 2) = mdocstart.x + 1200 + action.w - size
let POINT(2, 2) = mdocstart.y - 4600 + docum.h
let POINT(1, 3) = mdocstart.x + 1200 +action.w
let POINT(2, 3) = mdocstart.y - 4600 + docum.h + size
CALL FILLCOLOR.R(color)
CALL FILLSTYLE.R(1)
CALL FILLAREA.R (3, point(*, *))
call CLOSE.SEG.R
"end of right up (Good delivery)

"draw the arrowed link between the document and the action
call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
let POINT(1, 1) = mstart.x + action.w * 0.5
let POINT(2, 1) = mstart.y - 2300 - 4600 - action.h
let POINT(1, 2) = mdocstart.x + action.w
let POINT(2, 2) = mdocstart.y - 4600 * 2 + docum.h
call LINECOLOR.R (color)
call LINESTYLE.R(solid.line)
CALL LINEWIDTH.R (width.line.link)
call POLYLINE.R (2, POINT(*, *))
call CLOSE.SEG.R

"Draw the arrow
call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
let POINT(1, 1) = mdocstart.x + action.w
let POINT(2, 1) = mdocstart.y - 4600 * 2 + docum.h
let POINT(1, 2) = mdocstart.x + action.w - size
let POINT(2, 2) = mdocstart.y - 4600 * 2 + docum.h
let POINT(1, 3) = mdocstart.x + action.w
let POINT(2, 3) = mdocstart.y - 4600 * 2 + docum.h + size
CALL FILLCOLOR.R(color)
CALL FILLSTYLE.R(1)
CALL FILLAREA.R (3, point(*, *))
call CLOSE.SEG.R
"end of right bottom(payment_ack)

end " DRAW.EXPORTER

" ============================================================



routine DRAW.TRANSPORTER

" Draw line
call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 0)
let POINT(1, 1) = tstart.x
let POINT(2, 1) = tstart.y
let POINT(1, 2) = tstart.x
let POINT(2, 2) = tstart.y - 4600
call LINECOLOR.R (2)
call LINESTYLE.R(solid.line)
```

```
CALL LINEWIDTH.R (width.line)
call POLYLINE.R (2, POINT(*, *))
call CLOSE.SEG.R

" Draw action boxes
call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
let POINT(1, 1) = tstart.x - action.w
let POINT(2, 1) = tstart.y -  2300 + action.h
let POINT(1, 2) = tstart.x + action.w
let POINT(2, 2) = tstart.y -  2300 + action.h
let POINT(1, 3) = tstart.x + action.w
let POINT(2, 3) = tstart.y -  2300 - action.h
let POINT(1, 4) = tstart.x - action.w
let POINT(2, 4) = tstart.y -  2300 - action.h
call FILLCOLOR.R(4)
call FILLSTYLE.R(1)
call FILLAREA.R(4, POINT(*, *))
call CLOSE.SEG.R

call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
call TEXTALIGN.R(1, 1)
call TEXTCOLOR.R(14)
call TEXTSIZE.R(550)
call WGTEXT.R("Tx Goods", tstart.x, tstart.y - 2300 - 4600 * 0)
call CLOSE.SEG.R

"Draw documents
"draw the arrowed link between the document and the action
call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
let POINT(1, 1) = tdocstart.x - action.w
let POINT(2, 1) = tdocstart.y - docum.h
let POINT(1, 2) = tstart.x - action.w * 0.5
let POINT(2, 2) = tstart.y - 2300 + action.h
call LINECOLOR.R (color)
call LINESTYLE.R(solid.line)
CALL LINEWIDTH.R (width.line.link)
call POLYLINE.R (2, POINT(*, *))
call CLOSE.SEG.R

"Draw the arrow
call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
let POINT(1, 1) = tstart.x - action.w * 0.5
let POINT(2, 1) = tstart.y - 2300 + action.h
let POINT(1, 2) = tstart.x - action.w * 0.5 - size
let POINT(2, 2) = tstart.y - 2300 + action.h
let POINT(1, 3) = tstart.x - action.w * 0.5
let POINT(2, 3) = tstart.y - 2300 + action.h + size
CALL FILLCOLOR.R(color)
CALL FILLSTYLE.R(1)
CALL FILLAREA.R (3, point(*, *))
call CLOSE.SEG.R
"end of left up (Goods_delivery)

"draw the arrowed link between the document and the action
call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
let POINT(1, 1) = tstart.x + action.w * 0.5
let POINT(2, 1) = tstart.y - 2300 - action.h
let POINT(1, 2) = tdocstart.x + action.w
let POINT(2, 2) = tdocstart.y - 4600 + docum.h
call LINECOLOR.R (color)
call LINESTYLE.R(solid.line)
CALL LINEWIDTH.R (width.line.link)
```

```
call POLYLINE.R (2, POINT(*, *))
call CLOSE.SEG.R

"Draw the arrow
call OPEN.SEG.R
call GPRIORITY.R(SEGID.V, 1)
let POINT(1, 1) = tdocstart.x + action.w
let POINT(2, 1) = tdocstart.y - 4600 + docum.h
let POINT(1, 2) = tdocstart.x + action.w - size
let POINT(2, 2) = tdocstart.y - 4600 + docum.h
let POINT(1, 3) = tdocstart.x + action.w
let POINT(2, 3) = tdocstart.y - 4600 + docum.h + size
CALL FILLCOLOR.R(color)
CALL FILLSTYLE.R(1)
CALL FILLAREA.R (3, point(*, *))
call CLOSE.SEG.R
" end of right bottom (Good)

end "DRAW.TRANSPORTER

"  ============================================================


routine INIT
reserve POINT(*, *) as 2 by 4

"set world coordination
let XLO = -200       let V.XLO =  2500
let XHI = +200       let V.XHI = 30000
let YLO = -200       let V.YLO = 9200
let YHI = +200       let V.YHI = 29000

"set docu, action box, state ini coordination
let cstart.x = 7500         let cstart.y = 27000
let cdocstart.x = cstart.x + action.w
let cdocstart.y = cstart.y - 4600
let mstart.x = 15500      let mstart.y =  cstart.y -2300
let mdocstart.x = mstart.x - action.w
let mdocstart.y = mstart.y
let tstart.x = 24200        let tstart.y =  mstart.y - 2300
let tdocstart.x = tstart.x - action.w
let tdocstart.y = tstart.y

"constant values
let action.h = 800           let action.w = 2000
let docum.w = 500            let docum.h = 600
let solid.line = 1           let width.line = 100
let state.h = 400            let width.line.link = 20
let color = 2                let size = 300

"define the display doc and states positions
let gstate.x = -127            let gstate.y = 152
let cp1state.x = -128        let cp1state.y = 62
let cp1doc.x = -48             let cp1doc.y = 57
let cp2state.x = -128        let cp2state.y = -32
let cp2doc.x = -49             let cp2doc.y = -32
let cp3state.x = -127         let cp3state.y = -123
let mp1state.x = -12         let mp1state.y = 20
let mp1doc.x = 89            let mp1doc.y = 5
let mp2state.x = -12          let mp2state.y = -71
let mp2doc.x = 68             let mp2doc.y = -75
let tp1state.x = 115          let tp1state.y = -26
let tp1doc.x = 189            let tp1doc.y = -42

"define the transition and arrows
let cs.x = -127                 let cs0.y = 152
let cs1.y = 62                  let cs2.y = -32
```

```
let cs3.y = -123              let ts1.y = -26
let ms.x = -12                let ms0.y = 118
let ms1.y = 20                let ms2.y = -71
let ts.x = 115                let ts0.y = 70

"define the dosuments positions
let gdin.x = 76               let gdin.y = 80
let gout.x = 135              let gout.y = -27
let poin.x = -55              let poin.y = 122
let gdout.x = 35              let gdout.y = 19
let poout.x = -102            let poout.y = 65
let gin.x = -170              let gin.y = 75
let pain.x = -55              let pain.y = 32
let paout.x = -102            let paout.y = -20
let pacin.x = -170            let pacin.y = -15
let pacout.x = 15             let pacout.y = -60

call DRAW.BORDER
call DRAW.LABELS
Call DRAW.IMPORTER
call DRAW.EXPORTER
call DRAW.TRANSPORTER

let VXFORM.V = 1
call SETWORLD.R given XLO, XHI, YLO, YHI
call SETVIEW.R given V.XLO, V.XHI, V.YLO, V.YHI

"Display all state circles
display cs0 with "cstate0.icn" at (cs.x, cs0.y)
display cs1 with "cstate1.icn" at (cs.x, cs1.y)
display cs2 with "cstate2.icn" at (cs.x, cs2.y)
display cs3 with "cstate3.icn" at (cs.x, cs3.y)
display ms0 with "mstate0.icn" at (ms.x, ms0.y)
display ms1 with "mstate1.icn" at (ms.x, ms1.y)
display ms2 with "mstate2.icn" at (ms.x, ms2.y)
display ts0 with "tstate0.icn" at (ts.x, ts0.y)
display ts1 with "tstate1.icn" at (ts.x, ts1.y)

"Display Documents
display Doc3in with "gdin.icn" at (gdin.x, gdin.y)
display Doc4out with "gout.icn" at (gout.x, gout.y)
display Doc1in with "poin.icn" at (poin.x, poin.y)
display Doc3out with "gdout.icn" at (gdout.x, gdout.y)
display Doc1out with "poout.icn" at (poout.x, poout.y)
display Doc4in with "gin.icn" at (gin.x, gin.y)
display Doc5in with "pain.icn" at (pain.x, pain.y)
display Doc5out with "paout.icn" at (paout.x, paout.y)
display Doc6in with "pacin.icn" at (pacin.x, pacin.y)
display Doc6out with "pacout.icn" at (pacout.x, pacout.y)

end "INIT

" ============================================================
```

Appendix B

Second Implementation of GT3 in GEMA

Below is the pattern file in GEMA for process routines. This pattern file can translates user written structured statements into executable SIMSCRIPT II.5 source code automatically.

```
\Ipstep\W\(\W*\W,\W*\W,\W*\W,\W*\W,\W*\W,\W*\W,\W*\W,\W*\W,\W*\W\)=
\"(Processname, Nextstate, Outgoingdoc, Currentstate, Incomingdoc, NextstateXco,
NextstateYco, OutgoingdocXco, OutgoingdocYco)
\n"($1, $2, $3, $4, $5, $6, $7, $8, $9)\n
\"These arguments are needed by this specific process\n
\n\IProcess $1\n\n\'loop\'\n
\n\I@cmpn{$2;0;;;request 1 statetoken($2)
\ndisplay IMAGE$2 with "state$2white.icn" at ($6,$7)\n}
\n\I@cmpn{$3;0;;;request 1 doctoken($3)
\ndisplay Doc($3) with "whitedoc$3" at ($8, $9)\n}
\n\Iwait .TIME.UNIT unit\n\n
\I@cmpn{$5;0;;;request 1 doctoken($5)\n}
\n\I@cmpn{$4;0;;;request 1 statetoken($4)
\nwait .TIME.UNIT unit\n}
\n\I@cmpn{$2;0;;;relinquish 1 statetoken($2)
\ndisplay IMAGE$2 with "state$2black.icn" at ($6, $7)\n}
\n\I@cmpn{$3;0;;;relinquish 1 doctoken($3)
\ndisplay Doc($3) with "blackDoc$3" at ($8, $9)\n}
\n\I@cmpn{$5;0;;;relinquish 1 doctoken($5)\n}
\n\I@cmpn{$4;0;;;relinquish 1 statetoken($4)\n}
\n\Igo to \'loop\'\n
\nend\"$1\n\n
-----------------------------------------------------------------------------------------------------------------------------------------
```

Users can write the following structured statements as an input file for all process routines. We have introduced the parameters in detail in Chapter 5.

```
pstep (Generator, 1,-1, -1, -1, gstate.x, gstate.y, gdoc.x, gdoc.y)
"-------------------------------------------------------------------------------
pstep (CustomerP1, 2, 1, 1, -1, cp1state.x, cp1state.y,cp1doc.x, cp1doc.y)
"-------------------------------------------------------------------------------
pstep (CustomerP2, 3, 5, 2, 4, cp2state.x, cp2state.y,cp2doc.x, cp2doc.y)
"-------------------------------------------------------------------------------
pstep (CustomerP3, 4, -1, 3, 6, cp3state.x, cp3state.y,cp3doc.x, cp3doc.y)
"-------------------------------------------------------------------------------
pstep (MerchantP1, 6, 3, 5, 1, mp1state.x, mp1state.y, mp1doc.x, mp1doc.y)
"-------------------------------------------------------------------------------
pstep (MerchantP2, 7, 6, 6, 5, mp2state.x, mp2state.y,mp2doc.x, mp2doc.y)
"-------------------------------------------------------------------------------
pstep (TransporterP1, 9, 4, 8, 3, tp1state.x, tp1state.y,tp1doc.x, tp1doc.y)
"-------------------------------------------------------------------------------
```

The input file is translated by our pattern file into an output file automatically shown below. This output file can be compiled and executed in SimLab.

```
"(Processname, Nextstate, Outgoingdoc, Currentstate, Incomingdoc, NextstateXco, NextstateYco, OutgoingdocXco,
OutgoingdocYco)
"(Generator, 1, -1, -1, -1, gstate.x, gstate.y, gdoc.x, gdoc.y)
"These arguments are needed by this specific process

Process Generator
'loop'
request 1 statetoken(1)
display IMAGE1 with "state1white.icn" at (gstate.x,gstate.y)
wait .TIME.UNIT unit
relinquish 1 statetoken(1)
display IMAGE1 with "state1black.icn" at (gstate.x, gstate.y)
go to 'loop'
end"Generator
"-----------------------------------------------------------------------

"(Processname, Nextstate, Outgoingdoc, Currentstate, Incomingdoc, NextstateXco, NextstateYco, OutgoingdocXco,
OutgoingdocYco)
"(CustomerP1, 2, 1, 1, -1, cp1state.x, cp1state.y, cp1doc.x, cp1doc.y)
"These arguments are needed by this specific process
Process CustomerP1
'loop'
request 1 statetoken(2)
display IMAGE2 with "state2white.icn" at (cp1state.x,cp1state.y)
request 1 doctoken(1)
display Doc(1) with "whitedoc1" at (cp1doc.x, cp1doc.y)
wait .TIME.UNIT unit
request 1 statetoken(1)
wait .TIME.UNIT unit
relinquish 1 statetoken(2)
display IMAGE2 with "state2black.icn" at (cp1state.x, cp1state.y)
relinquish 1 doctoken(1)
display Doc(1) with "blackDoc1" at (cp1doc.x, cp1doc.y)
relinquish 1 statetoken(1)
go to 'loop'
end"CustomerP1
"-----------------------------------------------------------------------

"(Processname, Nextstate, Outgoingdoc, Currentstate, Incomingdoc, NextstateXco, NextstateYco, OutgoingdocXco,
OutgoingdocYco)
"(CustomerP2, 3, 5, 2, 4, cp2state.x, cp2state.y, cp2doc.x, cp2doc.y)
"These arguments are needed by this specific process
Process CustomerP2
'loop'
request 1 statetoken(3)
display IMAGE3 with "state3white.icn" at (cp2state.x,cp2state.y)
request 1 doctoken(5)
display Doc(5) with "whitedoc5" at (cp2doc.x, cp2doc.y)
wait .TIME.UNIT unit
request 1 doctoken(4)
request 1 statetoken(2)
wait .TIME.UNIT unit
relinquish 1 statetoken(3)
display IMAGE3 with "state3black.icn" at (cp2state.x, cp2state.y)
relinquish 1 doctoken(5)
display Doc(5) with "blackDoc5" at (cp2doc.x, cp2doc.y)
relinquish 1 doctoken(4)
relinquish 1 statetoken(2)
go to 'loop'
end"CustomerP2
"-----------------------------------------------------------------------

"(Processname, Nextstate, Outgoingdoc, Currentstate, Incomingdoc, NextstateXco, NextstateYco, OutgoingdocXco,
OutgoingdocYco)
"(CustomerP3, 4, -1, 3, 6, cp3state.x, cp3state.y, cp3doc.x, cp3doc.y)
```

```
"These arguments are needed by this specific process
Process CustomerP3
'loop'
request 1 statetoken(4)
display IMAGE4 with "state4white.icn" at (cp3state.x,cp3state.y)
wait .TIME.UNIT unit
request 1 doctoken(6)
request 1 statetoken(3)
wait .TIME.UNIT unit
relinquish 1 statetoken(4)
display IMAGE4 with "state4black.icn" at (cp3state.x, cp3state.y)
relinquish 1 doctoken(6)
relinquish 1 statetoken(3)
go to 'loop'
end"CustomerP3
"---------------------------------------------------------------------

"(Processname, Nextstate, Outgoingdoc, Currentstate, Incomingdoc, NextstateXco, NextstateYco, OutgoingdocXco,
OutgoingdocYco)
"(MerchantP1, 6, 3, 5, 1, mp1state.x, mp1state.y, mp1doc.x, mp1doc.y)
"These arguments are needed by this specific process
Process MerchantP1
'loop'
request 1 statetoken(6)
display IMAGE6 with "state6white.icn" at (mp1state.x,mp1state.y)
request 1 doctoken(3)
display Doc(3) with "whitedoc3" at (mp1doc.x, mp1doc.y)
wait .TIME.UNIT unit
request 1 doctoken(1)
request 1 statetoken(5)
wait .TIME.UNIT unit
relinquish 1 statetoken(6)
display IMAGE6 with "state6black.icn" at (mp1state.x, mp1state.y)
relinquish 1 doctoken(3)
display Doc(3) with "blackDoc3" at (mp1doc.x, mp1doc.y)
relinquish 1 doctoken(1)
relinquish 1 statetoken(5)
go to 'loop'
end"MerchantP1
"---------------------------------------------------------------------

"(Processname, Nextstate, Outgoingdoc, Currentstate, Incomingdoc, NextstateXco, NextstateYco, OutgoingdocXco,
OutgoingdocYco)
"(MerchantP2, 7, 6, 6, 5, mp2state.x, mp2state.y, mp2doc.x, mp2doc.y)
"These arguments are needed by this specific process
Process MerchantP2
'loop'
request 1 statetoken(7)
display IMAGE7 with "state7white.icn" at (mp2state.x,mp2state.y)
request 1 doctoken(6)
display Doc(6) with "whitedoc6" at (mp2doc.x, mp2doc.y)
wait .TIME.UNIT unit
request 1 doctoken(5)
request 1 statetoken(6)
wait .TIME.UNIT unit
relinquish 1 statetoken(7)
display IMAGE7 with "state7black.icn" at (mp2state.x, mp2state.y)
relinquish 1 doctoken(6)
display Doc(6) with "blackDoc6" at (mp2doc.x, mp2doc.y)
relinquish 1 doctoken(5)
relinquish 1 statetoken(6)
go to 'loop'
end"MerchantP2
"---------------------------------------------------------------------

"(Processname, Nextstate, Outgoingdoc, Currentstate, Incomingdoc, NextstateXco, NextstateYco, OutgoingdocXco,
OutgoingdocYco)
```

```
"(TransporterP1, 9, 4, 8, 3, tp1state.x, tp1state.y, tp1doc.x, tp1doc.y)
"These arguments are needed by this specific process
Process TransporterP1
'loop'
request 1 statetoken(9)
display IMAGE9 with "state9white.icn" at (tp1state.x,tp1state.y)
request 1 doctoken(4)
display Doc(4) with "whitedoc4" at (tp1doc.x, tp1doc.y)
wait .TIME.UNIT unit
request 1 doctoken(3)
request 1 statetoken(8)
wait .TIME.UNIT unit
relinquish 1 statetoken(9)
display IMAGE9 with "state9black.icn" at (tp1state.x, tp1state.y)
relinquish 1 doctoken(4)
display Doc(4) with "blackDoc4" at (tp1doc.x, tp1doc.y)
relinquish 1 doctoken(3)
relinquish 1 statetoken(8)
go to 'loop'
end"TransporterP1
"-------------------------------------------------------------------------
```

References

Allen, R. (2001). *Workflow: An Introduction*. Workflow Management Coalition. Available: http://www.wfmc.org/standards/docs.htm [2001, 07 Dec.].

Alonso, G., Agrawal, D., Abbadi, A. E., & Mohan, C. (1997). Functionality and Limitations of Current Workflow Management Systems. *IEEE Expert, 12*(5), pp. 68-74.

Bacceli, F., Cohen, G., Olsder, G. J., & Quadrat, J. P. (1992). *Synchronization and Linearity: An Algebra for Discrete Event Systems*. Chichester, England. Wiley.

Bons, R. W. H., Lee, R. M., Wagenaar, R. W., & Wrigley, C. D. (1995). Modelling inter-organizational trade using Documentary Petri Nets. *Proceedings of the Twenty-Eighth Hawaii International Conference on System Sciences.* IEEE, pp.189-198.

CACI. (1997a). *Building Simulation Models with SIMSCRIPT II.5*. CACI Products Company. Available: http://www.caciasl.com/simscript/simscript/simscript_docs.cfm [2001, 10 Jun.].

CACI. (1997b). *Getting Started with SIMSCRIPT II.5*. CACI Products Company. Available: http://www.caciasl.com/simscript/simscript/simscript_docs.cfm [2001, 10 Jun.].

Casati, F., Ceri, S., Pernici, B., & Pozzi, G. (1998). Workflow Evolution. *Data & Knowledge Engineering, 24*(3), pp. 211-238.

Clemons, E. K. (1995). Using Scenario Analysis to Manage the Strategic Risks of Reengineering. *Sloan Management Review, 36*(4), pp. 61-71.

Compatangelo, E., & Rumolo, G. (1997). Automated Reasoning about Enterprise Concepts. *ACM SIGGROUP Bulletin, 18*(2), pp. 56-58.

Curtis, B., Kellner, M. I., & Over, J. (1992). Process Modelling. *Communications of the ACM, 35*(9), pp. 75-90.

Davenport, T. H. (1993). *Process Innovation: Reengineering Work Through Information Technology.* Boston, USA. Harvard Business School Press.

Davenport, T. H., & Short, J. E. (1990). The New Industrial Engineering: Information Technology and Business Process Redesign. *Sloan Management Review,* 31(4), pp. 11-27.

Davenport, T. H., & Stoddard, D. B. (1994). Reengineering: Business Change of Mythic Proportions? *MIS Quarterly, 18*(2), pp. 121-127.

Davies, R., & O'Keefe, R. M. (1989). *Simulation Modelling with Pascal*. London, UK. Prentice Hall, Inc.

Delargy, K. M. (2001). The New Business Landscape. *Manufacturing Engineer, 80*(4), pp. 169-174.

Earl, M. J. (1994). The New and the Old of Business Process Redesign. *Strategic Information Systems, 3*(1), pp. 5-22.

Emshoff, J. R., & Sisson, R. L. (1971). *Design and Use of Computer Simulation Models*. New York, USA. Macmillan.

Galliers, R. D. (1993). Towards a Flexible Information Architecture: Integrating Business Strategies, Information Systems Strategies and Business Process Redesign. *Information Systems, 3*(3), pp.199-213.

Gordon, G. (1975). *The Application of GPSS V to Discrete System Simulation*. N.J., USA. Prentice-Hall.

Gray, D. N. (1995). *Gema -- A General Purpose Macro Processor*. Available: ftp://ftp.ugcs.caltech.edu/pub/gema/html/gema.html [2001, 20 Nov.].

Graybeal, W. J., & Pooch, U. W. (1980). *Simulation, Principles and Methods*. Cambridge, UK. Winthrop Publishers.

Hammer, M., & Champy, J. (1992). What is Reengineering? *Information week, 372*(5), pp. 10,14, 18, 20, 24.

Hammer, M., & Champy, J. (1993). *Reengineering the Corporation*. New York, USA. Harper Business.

Hammer, M., & Stanton, S. A. (1995). The Reengineering Revolution. *Government Executive, 27*(9), pp. 2A-7A.

Hiatt, J. (2001). *Business Process Reengineering Tutorial Series*. BPR Online Learning Centre. Available: http://www.prosci.com/mod1.htm [2001, 27 Nov.].

Hilpinen, R. (1971). *Deontic Logic: Introductory and Systematic Readings*. Miami, USA. Reidel Publishing Company.

Hlupic, V., & Robinson, S. (1998). Business Process Modelling and Analysis Using Discrete-Event Simulation. *Proceedings of the 1998 Winter Simulation Conference.* ACM Press, pp. 1363-1369.

Hoffman, D. L., & Novak, T. P. (1997). A New Marketing Paradigm for Electronic Commerce. *Information Society, 13*(1), pp. 43-45.

Jablonski, S. (1995). On the Complementarity of Workflow Management and Business Process Modelling. *ACM SIGOIS Bulletin, 16*(1), pp. 33-38.

Jackson, B. (1996). Reengineering the Sense of Self: The Manager and the Management Guru. *Journal of Management Studies, 33*, pp. 571-590.

Jensen, K. (1992). *Coloured Petri Nets*. New York, USA. Springer-Verlag.

Johnston, R. (1987). Complex Interactive Computer-Based Simulations Without Mathematical Modelling: A UK Experience. *Simulation and Games, 18*(4), pp. 501-516.

Kettinger, W. J., Teng, J. T. C., & Guha, S. (1997). Business Process Change: A Study of Methodologies, Techniques, and Tools. *MIS Quarterly, 21*(1), pp. 55-80.

Kindred, H. M. (1988). Modern Methods of Processing Overseas Trade. *Journal of World Trade, 22*(6), pp. 5-17.

Kleyn, M. F., & Browne, J. C. (1993). A High Level Language for Specifying Graph Based Languages and their Programming Environments. *Proceedings of the Fifteenth International Conference on Software Engineering.* ACM ICSE, pp. 324-335.

Law, A. M., & Kelton, W. D. (1991). *Simulation Modelling and Analysis* (2nd ed.) New York, USA. McGraw-Hill, Inc.

Law, A. M., & Larmey, C. S. (1984). *Introduction to Simulation Using Simscript II.5*. Calif., USA. CACI Products Company.

Lee, R. M. (1992). *CASE/EDI: EDI Modelling -- User Documentation*. Available:http://www.vwl.uni-freiburg.de/fakultaet/wi/docs/edipots.ps.gz [2001, 15 Feb.].

Lee, R. M. (1992). Dynamic Modelling of Documentary Procedures: A Case for EDI. *Proceedings of the Third International Conference on Dynamic Modelling of Information Systems*. IEEE, pp. 126-134.

Lee, R. M. (1998). INTERPROCS: A Java-Based Prototyping Environment for Distributed Electronic Trade Procedures. *Proceedings of the Thirty-First Hawaii International Conference on System Sciences.* IEEE, pp.202-209.

Lee, R. M. (1999). Distributed Electronic Trade Scenarios: Representation, Design, and Prototyping. *Electronic Commerce, 3*(Special Issue on Formal Aspects of Digital Commerce), pp. 105-136.

Lewis, C., Brooks, R. J., & Robinson, S. (2001). *Simulation* (1st ed.). New York, USA. Palgrave Publishers Ltd.

Mauw, S., & Reniers, M. A. (1994). An Algebraic Semantics of Basic Message Sequence Charts. *The Computer, 37*(4), pp. 269-277.

Meel, J. W. v., & Sol, H. G. (1996). Business Engineering: Dynamic Modelling Instruments for a Dynamic World. *Simulation and Gaming, 27*(4), pp. 40-61.

Mundie, D. A. (1996). *Why I Love Gema*. Available: http://www.anthus.com/Gema/WhyILoveGema.html [2002, 06 Jan.].

Murata, T. (1989). Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*. IEEE, 77(4), pp. 541-580.

Murata, T. (1992). Petri Nets Modelling and Analysis of Concurrent Systems. *Computer and Systems Science,* 44(3), pp. 28-39.

Pegden, C. D., Sadowski, R. P., & Shannon, R. E. (1990). *Introduction to Simulation using SIMAN*. Sewickley, Pa. USA. Systems Modelling Corporation.

Peterson, J. L. (1978). An Introduction to Petri Nets. *Proceedings of the National Electronics Conference.* National Engineering Consortium, Inc., pp. 144-158.

Peterson, J. L. (1981). *Petri Net Theory and the Modelling of Systems*. N.J., USA. Prentice-Hall.

Pidd, M. (1988). *Computer Simulation in Management Science* (2nd ed.). New York, USA. John Wiley & Sons.

Reisig, W. (1992). *A Primer in Petri Net Design*. New York, USA. Springer-Verlag.

Saxena, K. B. C. (1996). Reengineering Public Administration in Developing Countries. *Long Rang Planning, 29*(5), pp. 703-711.

Schal, T. (1996). *Workflow Management Systems for Process Organizations*. New York, USA. Springer-Verlag.

SDS. (2001, 13/08/2001). *What is the Relationship of System Thinking to System Dynamics?* System Dynamic Society. Available: http://www.albany.edu/cpr/sds [2001, 15 Dec.].

Sharp, A., & McDermott, P. (2001). *Workflow Modelling: Tools for Process Improvement and Application Development*. London, England. Artech House.

Sokol, P. K. (1995). *From EDI to Electronic Commerce: a Business Initiative*. New York, USA. McGraw-Hill, Inc.

Soloman, K. (2000). *Risk Management for Documentary Credits*. Australian Institute of Export (SA) Ltd. Available: http://www.aiex.com.au/soloman02.htm [2002, 29 Jan.].

Sommerville, I. (1992). *Software Engineering* (4th ed.). Workingham, England. Addison-Wesley Pub. Co.

Sophim. (2001). *Introduction to Electronic Data Interchange*. SOPHIM, Inc. Available: http://www.unedifact.com/edi.htm [2001, 25 Nov.].

Streng, R. J. (1993). *Dynamic Modelling to Assess the Value of Electronic Data Interchange.* PhD thesis, University of Delft, the Netherlands.

Suzuki, I. (1990). Formal Analysis of the Alternating Bit Protocol by Temporal Petri Nets. *IEEE Transactions Software Engineering, 16*(11), pp. 1273-1281.

Swami, A. (1995). Building the Business Using Process Simulation. *Proceedings of the 1995 Winter Simulation Conference.* ACM Press, pp. 1081-1086

Teng, J. T. C., Jeong, S. R., Kettinger, W. J., & Grover, V. (1995). The Implementation of Business Process Reengineering. *Management Information Systems, 12*(1), pp. 9-33.

Tsalgatidou, A., Louridas, P., Fesakis, G., & Schizas, T. (1996). Multilevel Petri Nets for Modelling and Simulating Organizational Dynamic Behaviour. *Simulation & Gaming, 27*(4), pp. 484-506.

Van der Aalst, W. M. P. (1992). *Timed Coloured Petri Nets and Their Application to Logistics.* PhD Thesis, Eindhoven University of Technology, the Netherlands.

Van der Aalst, W. M. P. (1994). Putting Petri Nets to Work in Industry. *Computer in Industry, 25*(1), pp. 45-54.

Van der Aalst, W. M. P. (1998). The Application of Petri Nets to Workflow Management. *Journal of Circuits, Systems and Computers, 8*(1), pp. 21-66.

Van der Aalst, W. M. P., & Heea, K. M. v. (1996). Business Process Redesign: A Petri-Net-Based Approach. *Computers in Industry, 29*(1-2), pp. 15-26.

Warren, J. R. (1996). Guest Editorial: Simulation of Information Systems. *Simulation and Gaming, 27*(4), pp. 438-439.

WFMC. (2001). *Workflow Management Coalition White Papers*. Workflow Management Coalition. Available: http://www.wfmc.org [2001, 15 Dec.].

Whatis. (2001, 30 Jul. 2001). *Whatis*. TechTarget. Available: http://whatis.techtarget.com [2001, 15 Dec.].

Willcocks, L., & Smith, G. (1995). *IT-enabled Business Process Reengineering: From Theory to Practice.* Oxford, England. Oxford Institute of Information Management, Templeton College.

Zhou, M., & DiCesare, F. (1993). *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems*. Boston, USA. Kluwer Academic Publishers.