Automating Supplier Selection Procedures

Reggie Davidrajuh
Narvik Institute of Technology
Narvik, Norway

Submitted to the Norwegian University of Science and Technology (NTNU) in partial fulfillment of the requirements for a Doctor of Engineering Degree.

Narvik, 2000

PREFACE

This work started on October 1997. The aim of the Ph.D. research was to find an optimal methodology and tools to automate the supplier selection procedures, especially for small and medium-sized enterprises, as these enterprises often lack in-house knowledge of achieving this. This dissertation solves the problem using a methodology consisting three components. The three components are a data collection system (based on mobile agents and XML) for information gathering, an inference engine (based on array-based logic) to make on-line decisions real-time, and a performance evaluation engine (based on Manufacturing systems theory and Petri net) to evaluate the performance of a supplier in real-time.

In the first chapter (introduction) the key concepts such as agile virtual enterprise, supply chain management, and small and medium-sized enterprises are introduced. At the end of this chapter, the problem statement of this Ph.D. research and the results of this research (significance) are stated. The second chapter is about a modeling approach for modeling supplier selection procedures.

The third chapter introduces the methodology for automating supplier selection procedures. The fourth, fifth and sixth chapters are about the components of the methodology for automation. The components are the data collection system, the inference engine, and the performance evaluation engine respectively.

I would like to thank professors Ziqiong Deng and Øyvind Bjørke for their useful advice and helps. My thanks goes to the library staff at Narvik Institute of Technology for their kind cooperation. Finally, I owe a big thank you to my wife Ruglin Romeula for her love and patience.

Reggie Davidrajuh
Narvik, 18. December 2000

ABSTRACT

This dissertation describes a methodology, tools, and implementation techniques of automating supplier selection procedures of a small and medium-sized agile virtual enterprise.

Firstly, a modeling approach is devised that can be used to model the supplier selection procedures of an enterprise. This modeling approach divides the supplier selection procedures broadly into three stages, the pre-selection, selection, and post-selection stages.

Secondly, a methodology is presented for automating the supplier selection procedures. The methodology first divides the selection process into three stages, such as pre-selection, selection, and post-selection stages. Then the methodology identifies the steps within the selection stage that can automated. Automating the steps within the pre-selection and the post-selection stages are not considered here. The methodology also proposes three modules for automating the selection stage. To assist automation, the selection stage is further divided into three sub-stages, namely, bidder selection, partner selection and performance evaluation sub-stages. By this methodology, each sub-stage of the selection stage employs a module for automation; the modules used for automation at these sub-stages are the data collection system, the inference engine and the performance evaluation engine.

Thirdly, modeling, design and implementation of the three modules for automation are described:

The data collection system is for automating on-site selection sub-stage. The data collection system sends mobile agents to collects data (or quotes) from suppliers' web sites; to enable this, data provided by suppliers on their web sites must be structured-information using XML conforming to a publicly available uniform grammar. The mobile agents then accepts the supplier as a potential supplier and bring backs the data to the main assembler for further scrutiny, only if the supplier data satisfies the critical condition sets by the assembler with broad margins.

The inference engine is the second module for automation, which is used for selecting the best supplier from a list of potential suppliers; this is done in the sub-stage partner selection. Array-based logic is used for realizing the inference engine because, array-based logic offers fast computation, compact code, and complete solution.

The last module for automating supplier selection procedures is the performance evaluation engine. The performance evaluation engine is a simple yet effective tool for performance evaluation of the supplier that is selected as the best supplier by the inference engine. The performance evaluation engine put the new supplier in collaboration with the other existing collaborating enterprises for a specific project, and carry out simulations to see whether the supplier will perform satisfactorily in collaboration. Realization of the performance evaluation engine is done in three stages. The first stage is the modeling stage where the manufacturing systems theory approach is used to establish a higher-level model of the agile virtual enterprise. The second stage is the simulation stage, where the higher-level model is converted into the lower-level Petri net model. The third stage is the implementation stage at which the lower-level Petri net model is converted to C++ programming language code and compiled into a executable system - the performance evaluation engine. To model the lower-level we make use of timed colored Petri net.

CONTENTS

1. INTRODUCTION

The title of this dissertation is "**automating supplier selection procedures for small and medium-sized agile virtual enterprises**". Supplier selection is an important part of supply chain management. In this introductory chapter, firstly, the key concepts such as *agile virtual enterprise*, *small and medium-sized enterprise* (SME), *supply chain management* are explained; *supplier selection procedures* are explained in the next chapter. Secondly, the main problem that is solved in this dissertation is presented. Finally, significance of this research is presented.

1.1 INTRODUCING THE KEY CONCEPTS

In the beginning of the eighties, competition in the market increased enormously, because of the increased customer awareness of the products and due to the open economic policies of the nations. New concepts evolved and tried to meet the new challenges; concepts like computer integrated manufacturing, lean manufacturing, agile manufacturing, and lately, agile virtual management practices or agile virtual enterprise.

1.1.1 A short historical excursion from CIM to agile virtual enterprise

Computer integrated manufacturing (CIM) combines the activities such as computer aided design (CAD), computer aided planning (CAP), computer aided manufacturing (CAM), computer aided quality control (CAQ), and production planning and control (PP&C) in one system [Rembold et al, 1993]. The combination is an *internal integration* within an enterprise. During the period of development and implementation of CIM, the Japanese employed just-in-time (JIT) concept to minimize costs; the JIT concept is about production where delivery of parts and materials for production happens in exact amount, at the correct time, in right quality. Also, production uses a minimum amount of time, facilities, equipment, materials, and human resources [Deng, 1997]. Zero inventory cost in JIT due to arrival of materials at the correct time means JIT depends on the interaction between the suppliers and the purchaser; thus, as opposed to CIM, JIT is no longer an internal issue ('intra-enterprise') but an *'inter-enterprise'* issue. JIT concepts are part of the wider *lean* production practices, invented by Toyota to produce high quality automobiles, cheaper and faster. The principles of lean production included teamwork of dedicated team players (unskilled workers in *mass production*, practiced by the European and American counterparts at that time), communication at the horizontal levels to confront conflicts in the early stage (no horizontal communication in mass production), efficient use of resources and elimination of waste, and continuous improvement [Womak et al, 1991]. Lean production incorporated supply and distribution chains which posses first signs of fuzzy business boundaries (*virtual enterprise*) and *agility*.

Agile manufacturing and virtual enterprise concepts were formally presented, for the first time in [Nagel et al, 1991]. Nagel et al (1991) and Goldman (1993) emphasized agility or the ability to react quickly as the central factor for survival of an enterprise, and to incorporate entire supply base and customers into a virtual corporation. Thus, *agile virtual enterprise* is based on two concepts; *agility* is the ability to react quickly to changing conditions; *virtual enterprise* means the ability to have fuzzy business boundaries for a project, extending the enterprise to include newer collaborating enterprises [Goranson, 1999].

With the advent and widespread use of Internet, internet-based applications are now used to improve competitiveness of the agile virtual enterprises. This new applications, known as *e-commerce* (electronic commerce) applications, enable an agile virtual enterprise to not only retain its distinctive partnership role in a collaborative manufacturing environment, but also build up its responsive production by delivering creative, timely and quality products to the global market [Yam et. al., 2000].

1.1.2 Introduction to agile virtual enterprise

Figure-1.1 shows both new and old, supply and distribution chain of an agile enterprise. In figure-1.1, the main producer or assembler integrates a number of collaborating enterprises (suppliers and distributors), to manufacture a certain class of product. When market conditions change, a new class of product or an improved version of the product should be quickly turned out to meet the new market requirements; that is, the main assembler has to be agile. In this case, the main assembler may seek for a new combination of suppliers and distributors that are more suitable to manufacture the new class of products [Deng, 1994; Davidrajuh and Deng, 1998].

As shown in figure-1.1, our view of the collaboration is 'main assembler-centered'. This means, the main assembler- the enterprise that owns the trademark of the product being produced by collaboration, decides whether to change the collaborating enterprises (accept any new enterprise into collaboration or to reject any existing collaborating enterprises), change the volume and properties of the product, etc.

The life cycle phases of agile virtual enterprise

The life cycle of agile virtual enterprise includes phases such as *business opportunity identification*, *partner selection*, *formation*, *operation*, and *reconfiguration* [Enator, 1998]. Before forming an agile virtual enterprise, profitability of a product (that is going to be produced) has to be assessed. This is done in the opportunity identification part of the formation phase. Profitability of a product is assessed by extensive market analysis and research. After the opportunity identification part, the right collaborating enterprises must be found to manufacture the product; thus the supply chain is established at this stage.

After forming an agile virtual enterprise, the collaboration is put to use for producing a class of products; this is the operation phase. During the operation phase and after, the main assembler constantly monitor the performance of the supply chain so as to determine whether to establish a new supply chain for producing the same class of product or a new class of product, by changing the collaborating enterprises; this phase is called the reconfiguration phase. Clearly, formation and reconfiguration phases are similar. In reconfiguration phase, the performance of an existing collaborating enterprise is compared with a new potential supplier.

1.1.3 Supply chain management in agile virtual enterprise

The supply chain management is an important part of an enterprise's strategy to optimize planning and execution processes to respond to the changes in the market. It is not an exaggeration to say that supply chain management has now become more important than the manufacturing processes themselves [Yam et. al., 2000].

Figure-1.1 shows the traditional supply chain management, spanning the full supply chain, from procuring raw materials from the raw materials suppliers to the delivering the finished good to the consumers; though the supply chain in figure-1.1 shows only two layers (two tiers) of suppliers and distributors, in reality there will be many layers.

There exist many definitions for the term supply chain management depending on the approach of the respective firm or branch. Germany's Fraunhofer Institute (IPA) defines Supply Chain Management (SCM) as the integrated planning, handling, co-ordination and management of material and information flows in single-level or multi-level supply chains [Sihn et al, 2000]. In this context, Supply Chain Management requires the extended cooperation of all partners in the supply chain in the following areas: ordering, forecasting, transport, material requirements planning, resource planning and scheduling. This cooperation can be achieved through the process-oriented reorganization of the supply chain and the introduction of the appropriate IT infrastructure, control system or other management principles [Sihn et al, 2000].

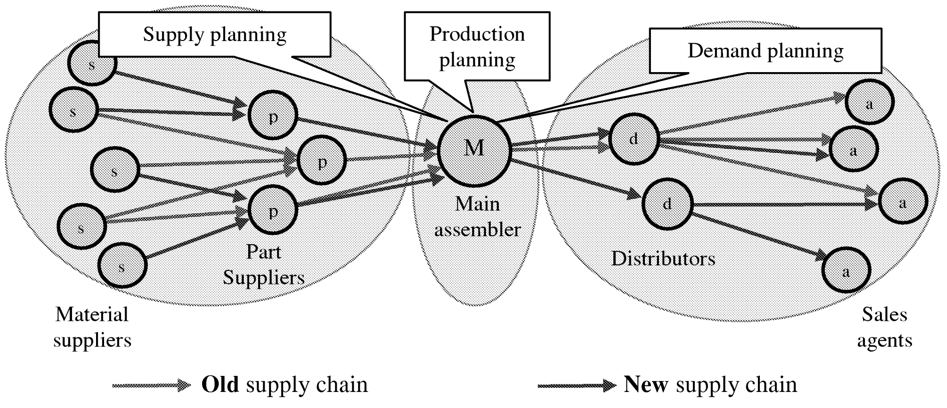


Figure-1.1: Simplified supply management system for a virtual enterprise

SCM is truly a multidisciplinary subject, evolving from diverse subjects like internal supply chaining [Harland, 1996]; business re-engineering [Dale, 1994; Croom et al, 2000]; operations management [Slack et al, 1998]; logistics and transportation and network [Handfield and Nichols, 1999]; marketing; organization behavior, industrial organization, systems engineering [Towers, 2000; Stevens, 1989].

1.1.4 Why supply chain management is important?

Supply chain management is very important for an agile virtual enterprise because, by proper supply chain management practices, current and future profitability of an enterprise is maximized. The following works [Christopher, 1993; Christopher,1998; Kjenstad, 1998; Croom et al, 2000; Fingar et al, 2000] cites many examples of supply chain management strategies whereby billions of dollars were saved (earned).

Though supply chain management is about strategically managing the procurement, movement and storage of materials, and the related information flow, in this dissertation-we shall concentrate only on one topic within supply chain management namely, the supplier selection procedures. The supplier selection procedures is the topic of the next chapter (chapter 2).

1.2 SMALL AND MEDIUM-SIZED ENTERPRISES

In general, small and medium-sized enterprises (SMEs) are classified as enterprises carrying out small to medium-scale manufacturing, employing fewer than 500 persons, and an annual turnover of £20 million [Gunasekaran, 2000]. The aim of this dissertation is to assist SMEs to automate supplier selection procedures mainly because of the four attributes attached to SMEs:

1. SMEs are the most important portion of the economy of any nation.
2. SMEs are flexible and innovative, taking into account the size and business structure
3. SMEs primarily satisfy local market
4. SMEs can not afford expensive application packages

1.2.1 SMEs are the largest component of advanced economies

In Germany, out of the 2.1 million enterprises, 99.8% employ less than 500 people and 94.7% less than 20 people [Bundesamt, 1987] as quoted in [Engelhardt, 1997]. In the UK economy, the influence of manufacturing businesses with less than 250 employees has been steadily increasing over the last twenty years [Storey, 1994]. In the USA, there were more than 5.7 million businesses in 1992, out of them, only 14 000 had more than 500 workers [US Small Business Administration, 1992].

1.2.2 SMEs are flexible and innovative

SMEs are flexible and innovative taking into account the size and business structure. The cost structure of an SME is different from larger enterprises; SME production is characterized by relatively high wages costs, but they spend less (compared to larger enterprises) on raw material, semi-finished products, and other products and services supplied to them [Gunasekaran, 2000]. Thus, one of the primary goals of the strategies adopted by SMEs is to be flexible on the procurements in-order to save as much as possible on the supplies without compromising quality and delivery time aspects. Because of their small size, the autonomous agents in-charge of procurement, can be more agile (without time-consuming internal consultations) in responding to newer opportunities, e.g. seeking suppliers with better bids.

1.2.3 SMEs primarily satisfy local market

SMEs generally manufacture more for local markets, of items such as consumer goods, investment goods, and suppliers' products and services [Gunasekaran, 2000]. In Australia, for example, only 11 percent of SMEs exported in 1995 [Yellow Pages Australia, 1995a; quoted in [Graham, 1999]. This implies, the supplier selection procedures for seeking optimum suppliers around the globe is much more important than seeking distributors overseas, for SMEs.

1.2.4 SMEs can not afford expensive application packages

SMEs can't always afford expensive SCM software like products from Baan, JD Edwards, SAP, PeopleSoft, i2 Technologies or Manugistics. SME need inexpensive but effective and powerful software to cooperate in their supply chain.

1.3 PROBLEM STATEMENT

The aim of this dissertation is three-folded:

- The first one is to find *a methodology* to automate supplier selection procedures.
- The second aim is to find or develop *optimal tools* for automating supplier selection procedures.
- The third aim is to conduct research leading to *realization* of a system or systems that can be implemented to pursue potential suppliers automatically.

1.4 SIGNIFICANCE OF THIS RESEARCH

This research involves applied mathematics (discrete mathematics and mathematical logic), distributed/applied computing (mobile agents, Java, CORBA, and XML) and management issues, thus a good example for cross-disciplinary research within industrial engineering. Considering the four flows that exist between collaborating enterprises in a virtual enterprise (the four flows are information, material, work, and fund), this research is about information flow and material flow.

This dissertation consists of some surveys, many algorithms for realizations of newer tools or improvement of existing tools. During the course of this Ph.D. research, two tools were developed that have potentials for many applications. More details are given below.

1.4.1 Why this research is necessary?

The Internet is bringing profound change to the business world and has enabled new ways of conducting business; to compete in the emerging digital economy, enterprises will need to change their business models, rethink the way they work and form new relationships with their collaborating enterprises. Even *the way* the new relationships are formed need to be changed to gain competitive advantage; this research is about how the new relationships (through supplier selection) should be established in an agile virtual environment using the global business channel - the Internet.

The Internet or World Wide Web precisely, allows enterprises from the smallest enterprise to largest corporations to establish global presence. Hence, small enterprise (SMEs) now have the opportunity to reach geographically dispersed markets that would otherwise been cost prohibitive to consider. SMEs also now have the opportunity to select the best suppliers, by utilizing suppliers' bids on WWW, thus averting time consuming and costly outside sources (or middlemen) like professional contacts, trade journals, directories, and import brokers. By best supplier, we mean the supplier who can supply an SME with right amount of material at the right time - thus preventing physical inventories at the SME, at right price (or cheap) of the right quality (for example, by adhering to ISO standards).

What we have said so far, is that using WWW for supplier selection is very important for survival of an SME. But this research goes further ahead; what we are going to say and show how is that, *automating* the supplier selection utilizing WWW is highly beneficial for an SME.

Automating the supplier selection procedures are very important for three reasons:

1. The first reason is that the volume of data that is available on the Internet is simply too much for manual processing; there must fully or semi- automated system to collect and process the huge amounts of data available on hundreds or even thousands of potential suppliers web sites.
2. The second reason is that, Internet enables *business ubiquity*, allowing an enterprise to conduct business all the time; certainly an autonomous system would be preferential than humans to work on a 24x7x365 basis.
3. The third reason is that, as we have already stated- SMEs with restricted revenue and production characterized by relatively high wages costs, can not afford further labor costs for processing the enormous data from suppliers. Therefore, this research was initiated to find a methodology, tools, and implementation of automating supplier selection procedures.

1.4.2 In what sense this research is new and what are the achievements?

Extensive literature study shows that automating supplier selection (fully or partially) is not done elsewhere. Therefore, the idea of automating supplier selection itself is new. The main contribution of this research can be summarized as follows:

- We have proposed and evaluated a methodology for automating supplier selection procedures addressing the requirements like agility, reliability, extensibility, interoperability and legacy integration (chapters 2 and 3). The methodology is divided into different phases reflecting the current supplier selection practices, also introducing new trends to support automation. The methodology provides an automation system consisting of three engines for data collection, inference, and performance evaluation respectively.

  By the methodology, the proposed system is cheap to build, can be build and implemented quickly, and require few staff to maintain. The proposed system is also scalable, and provides possibilities to coordinating with the rest of the supplier selection procedures that are not automated.

- We have proposed a data collection system that make use of WWW, based on XML and mobile agents (chapter 4). We have surveyed the enabling technologies most suitable for realizing the data collection system. We devised the data collection systems so as to satisfy the requirements such as portable, scalable, extensible, secure, affordable, and qualitative. The data collection system searches data from supplier autonomously, with no human intervention from the management, and frequently. The system is able to handle RFP (request for proposals) autonomously. The system is also self-starting and self-correcting. We made a prototype for the data collection system for the operation phase of an agile virtual enterprise where the main assembler uses the data collection system as the information infrastructure for collaboration with the other enterprises. We have also shown how to make use of the legacy systems efficiently.

- We have proposed an inference engine for making decisions based on the data collected from the potential suppliers (chapter 5). The inference engine has the qualities such as fast processing time for online real-time operation; compact size so that it does not require large memory, disk or extra processor; easy to build and maintain. We have developed a toolbox of logic functions based on array-based logic, which we call structured array-based logic. This toolbox allows direct implementation of the inference engine from the modeling and simulation stage, as well as is fast, compact, and complete.

  We developed the toolbox structured array-based logic with MATLAB with the aim of computing with words, in-addition to fast, compact and complete computation.

- Numerous mathematical methods are available for measuring the performance of a supplier, and the supply chain as a whole. However, we devised a simple yet effective method for performance measurement of a supplier, and of the whole supply chain. This method is based on manufacturing systems theory and Petri Net; see chapter 6.

- The proposed methodology for modeling and simulation supply chain using Petri nets is new concept (chapter 6). Modeling and simulation of supply chain using Petri nets is not done elsewhere[1]. We have also devised a set of new firing rules for the timed colored Petri net for this purpose. In addition, we developed two toolboxes of function for realizing the performance evaluation engine from the Petri net model. The toolboxes are AgileSIM and PenSIM, which run on MATLAB simulation system.

  AgileSIM is a toolbox of functions for modeling supply chain in a higher-level model (based on manufacturing systems theory approach) and then for yielding a lower-level Petri net model. PenSIM is a simulator based on the timed colored Petri net, for simulating lower-level Petri net model.

[1] according to discussions in the Petri nets Email List PetriNets@daimi.au.dk, dated 12 October 00.

2. MODELING SUPPLIER SELECTION PROCEDURES

The selection of suppliers is the responsibility of the purchasing enterprise and requires a consideration of several factors. Some enterprises employ simple procedures with few criteria for supplier selection, while others use complex procedures with many criteria divided into many categories. The complexity of the selection process depends on the size, business type and revenue of the purchasing enterprise, the total costs involved in purchasing, and on the fact that how often the purchase is to be repeated, etc.

The aim of this chapter is to develop a generic modeling approach for modeling supplier selection procedures, so that in the next chapter, this modeling approach can be used to identify the steps of supplier selection procedures that can be automated.

2.1 SOME FACTORS IN MODELING SUPPLIER SELECTION

Supplier selection is complicated by the fact that various criteria must be considered in the decision making process. The analysis of criteria for selection and measuring the performance of the suppliers has been the focus of many research papers; see for example [Weber et al, 1991], which reviews a total of 74 research papers on supplier selection.

In this section, first we shall go through the most important criteria for supplier selection, and then on the issues on the performance measurement of suppliers.

	Supplier selection criteria	*No. of research papers*	*%*
1.	**Net price**	**61**	**80**
2.	**Delivery**	**44**	**58**
3.	**Quality**	**40**	**53**
4.	Production capability	23	30
5.	Geographic location	16	21
6.	Technical capability	15	20
7.	Management and organization	10	13
8.	Reputation and position in industry	8	11
9.	Financial position	7	9
10.	Performance history	7	9

Table-2.1: Supplier selection criteria sited in various research papers [adapted from Weber et al, 1991]

2.1.1 The selection criteria

There are many criteria for supplier selection; since the first extensive review on supplier selection criteria by [Dickson, 1966], and then by [Weber et al, 1991] and the most recent ones [Chick et al, 2000; Ghingold and Wilson, 1998; Motwani et al, 1999] discusses the most important selection criteria which remain almost invariant.

Table-2.1 lists the top ten most important criteria discussed in numerous research papers. Table-2.1 lists each criterion together with number of research papers which embrace the criterion as well as the rank given to the criterion. Table-2.1 indicates that the (net) price, delivery (time) and quality as the most important supplier selection criteria, as these criteria were sited in 80%, 58% and 53% of the research papers. Consequently, our supplier selection modeling approach, and our methodology for automating supplier selection, are based on these three most important criteria only; this limitation is only for brevity. The ideas discussed in this dissertation can be extended to include other selection criteria too.

2.1.2 Measuring the performance of suppliers

When measuring the performance of a supplier, there are some performance factors that can be quantified or evaluated in monetary terms (like cost of product, delivery delay costs etc.); these quantifiable factors can be used in mathematical equation to measure the overall performance of a supplier.

There exist some factors that can not be quantified or can not be quantified easily (e.g. goodwill or reputation and position of an enterprise in industry). These factors can be still used for performance measurement if the techniques that deals with computations with words are employed. Fuzzy logic is a technology that computes factors that can not be quantified easily; structured array-based logic is yet another technology that deals with computing with words; both fuzzy logic and structured array-based logic are introduced in chapter-5.

In some literature (e.g. [Housshyar and Lyth, 1992]), quantifiable factors are termed objective factors and non-quantifiable factors are termed subjective factors.

2.2 A MODELING APPROACH

Several research studies attempt to develop models of supplier selection procedures for various reasons; for reviews, see [Chick et. al, 2000; Ghingold and Wilson, 1998; Motwani et al, 1999; Woodside et al, 1999; Yousef, 1992; Yousef et al, 1996]. Developing a generic model for the supplier selection procedure is not an easy task, because:

1. *It is multiple-person activity*: Supplier selection involves persons at several authority levels (vertical involvement), and across several departments (lateral involvement) within the purchasing enterprise.
2. *Type of procurement*: Supplier selection procedures vary for procurement of capital equipment and for commodities and MRO (maintenance, repair and operating) items.

3. *Duration of collaboration*: Supplier selection criteria depends on the duration of expected collaboration between the supplier and the purchaser; from short term commitment to long term alliance.
4. *Type of collaboration*: Selection criteria also depends on the closeness of collaboration between the supplier and the purchaser. Extensiveness of the selection criteria will be high if the potential supplier is to become a strategic partner.

Though several research studies have attempted to model supplier selection procedure, they are mostly about buying capital equipment, and about the behavior of the professional buyers; these research studies are not about creating a model of the supplier selection procedure *that is suitable for utilizing the Internet technology for automating supplier selection procedures (e.g. as an e-commerce application)*. In this section, we attempt to create a generic modeling approach by going through the literature study first; the modeling approach is developed by examining the complete buying process in an abstract manner, but general enough with regard to the dynamic nature of supplier selection and its implications for automating it. In the next section, we shall go through some case studies to verify whether our generic modeling approach is general enough to model the procedures utilized by the enterprises.

2.2.1 The modeling approach

We start with a broad classification of supplier selection procedures. Let us begin with a supplier selection procedure broadly categorized into the following three stages:

1. *Pre- Selection stage*: Management sets the strategic goals for procurement,
2. *Selection stage*: The main selection procedures, starting with many potential suppliers and ending with a most preferred supplier.
3. *Post Selection stage*: Establishing collaboration with the selected supplier.

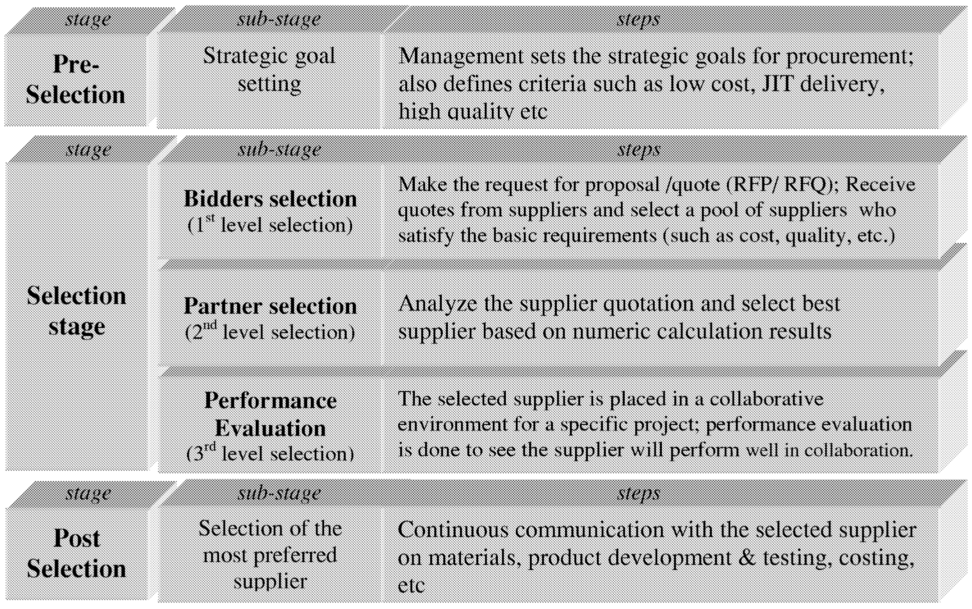
The modeling approach is shown in figure-2.1. The modeling approach distinctly partition the supplier selection procedures into the pre-selection, selection and post-selection stages. This is because, we want to separate the steps of the supplier selection procedures that can be automated from the steps that are best left for human intervention.

The pre-selection stage and the post-selection stage are mainly about managerial issues that are subjective, therefore should be left to humans rather than machines. Only the steps that fall within the selection stage are considered for automation in this dissertation. The methodology, and modules for automating the steps are discussed in chapter-3.

2.2.2 Verifying the modeling approach

In this subsection, first we shall study some models of supplier selection procedures that exist in literature; these models emphasize different aspects of supplier selection. By studying these models, we could verify whether our modeling approach (shown in figure-2.1) is general enough to create the models containing the main components emphasized in the these models.

For brevity, we limit ourselves to three models only; we could confidently state that the these three models are representative as numerous other models developed in many literature vary only slightly from these three models.



stage	*sub-stage*	*steps*
Pre-Selection	Strategic goal setting	Management sets the strategic goals for procurement; also defines criteria such as low cost, JIT delivery, high quality etc

stage	*sub-stage*	*steps*
Selection stage	**Bidders selection** (1st level selection)	Make the request for proposal /quote (RFP/ RFQ); Receive quotes from suppliers and select a pool of suppliers who satisfy the basic requirements (such as cost, quality, etc.)
	Partner selection (2nd level selection)	Analyze the supplier quotation and select best supplier based on numeric calculation results
	Performance Evaluation (3rd level selection)	The selected supplier is placed in a collaborative environment for a specific project; performance evaluation is done to see the supplier will perform well in collaboration.

stage	*sub-stage*	*steps*
Post Selection	Selection of the most preferred supplier	Continuous communication with the selected supplier on materials, product development & testing, costing, etc

Figure-2.1: Our modeling approach for modeling supplier selection procedures

The C&N six-phase model

A six-phase model identifying several steps of the supplier selection procedure is discussed in [Carter and Narashiman, 1990]; the C&N six-phase model focuses on a purchasing cycle that takes into account *international purchasing*. The six phases of the model is shown in figure-2.2.

Let us re-model the supplier selection processes described by the C&N six-phase model by our modeling approach. The resulting model obtained by our modeling approach is shown in tabular form in table-2.2; we call this model "the modified C&N model".

Let us compare the C&N six-phase model shown in figure-2.2 and the modified C&N model shown in table-2.2. These two models has the same steps; however, modified C&N model clearly illustrates the pre-selection, selection and post-selection stages, whereas in the C&N six-phase model the division is not visible. Therefore, after re-modeling by our modeling approach, the new model now shows the steps within the selection stage that are now ready for automation; how the automation is achieved is further discussed in chapter-3.

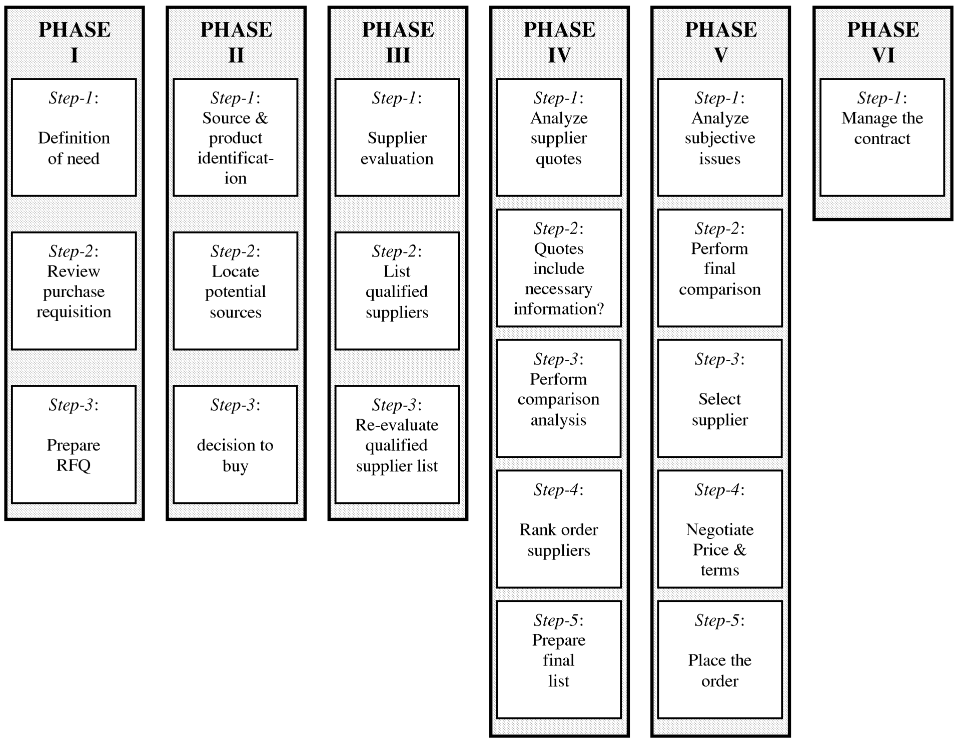

PHASE I
Step-1: Definition of need
Step-2: Review purchase requisition
Step-3: Prepare RFQ
PHASE II
Step-1: Source & product identificat-ion
Step-2: Locate potential sources
Step-3: decision to buy
PHASE III
Step-1: Supplier evaluation
Step-2: List qualified suppliers
Step-3: Re-evaluate qualified supplier list
PHASE IV
Step-1: Analyze supplier quotes
Step-2: Quotes include necessary information?
Step-3: Perform comparison analysis
Step-4: Rank order suppliers
Step-5: Prepare final list
PHASE V
Step-1: Analyze subjective issues
Step-2: Perform final comparison
Step-3: Select supplier
Step-4: Negotiate Price & terms
Step-5: Place the order
PHASE VI
Step-1: Manage the contract

Figure-2.2: The C&N six-phase model for supplier selection

A multi-agent model

Li et al (2000) proposes a *multi-agent model for partner selection in virtual enterprises*; a modified (simplified) version of this model is shown in figure-2.3.

In this multi-agent based supplier selection process, after the goals of the virtual enterprise are identified, the virtual enterprise coordinator agent (VCA) decomposes the goal into sub-goals so that individual enterprise agents (EAs) can receive these sub-goals as requests (or requests for bids). When the EAs respond with bids, these bids are evaluated using distributed constraint satisfaction techniques, so that a single most qualified supplier results.

Let us divide the multi-agent model into sectors so that it conforms to the three selection stages of our model; the re-worked model is shown in figure-2.4. It is notable that in the model shown in figure-2.4, the selection sub-stage performance evaluation is missing, despite our modeling approach indicates the necessity for that sub-stage. We may assume that this sub-stage too is included in the post-selection stage.

stage	*sub-stage*	*C&N Phases*	*steps*
Pre-selection	Strategic goal setting	***I***	1. Definition of need
			2. Review purchase requisition
Selection stage	Bidders selection (1st level selection)	***I/II/III***	1. Prepare RFQ
			2. Source and product identification
			3. Locate potential sources
			4. Decision to buy
			5. Supplier evaluation
			6. List qualified suppliers
	Partner selection (2nd level selection)	***III/IV***	1. Re-evaluate qualified supplier list
			2. Analyze supplier quotes
			3. Make sure quotes include necessary information
			4. Perform comparison analysis
			5. Rank order suppliers
			6. Prepare final list
	Performance evaluation (3rd level selection)	***V***	1. Analyze subjective issues
			2. Perform final comparison
			3. Select supplier
			4. Negotiate prices and terms
			5. Place orders
Post-selection	Relationship maintenance	***VI***	1. Manage the contract

Table-2.2: The modified C&N model with emphasis on international purchase

We believe that it is correct to say that by constraint satisfaction problem, the best set (of supplier quotes) can be selected out of many competing sets; but by only placing the winning supplier together with the other existing collaborating partners and measure the performance of the selected supplier in collaboration, we can be sure that the selected supplier will perform optimally as a new partner in the collaboration. In our modeling approach, the measurement of performance of the selected supplier in collaboration is done at the sub-stage performance evaluation of the supplier selection stage, which is a crucial part of the supplier selection stage for any virtual enterprise. However, many research papers (including [Li et al, 2000]) overlook this stage.

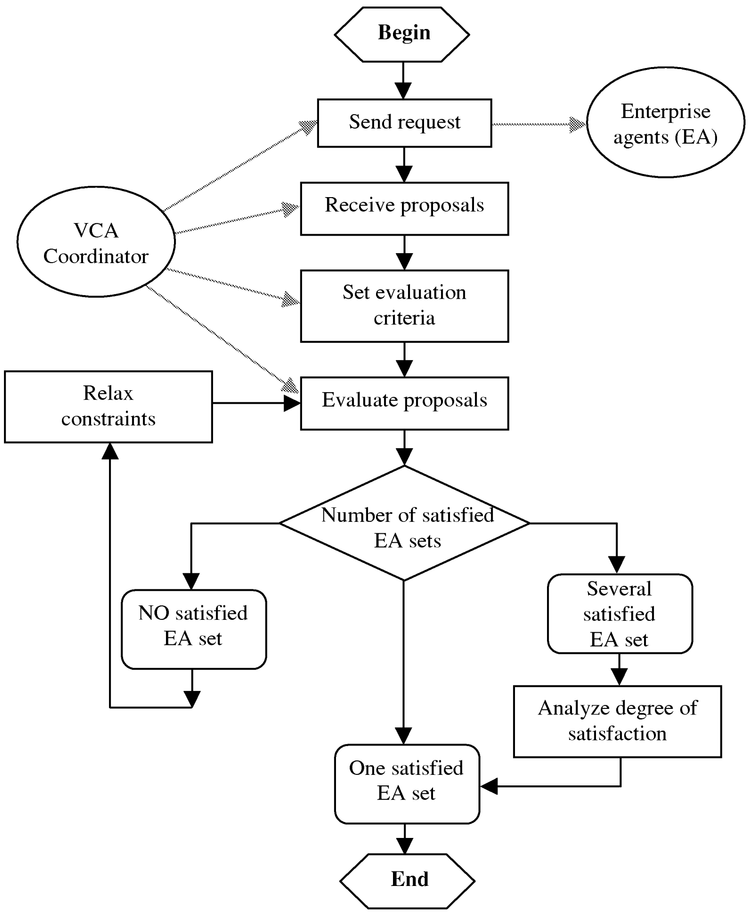


Figure-2.3: The multi-agent model for partner selection

Multi-attribute model for configurable machining system selection
Chick et al (2000) presents a multi-attribute model for selection of complex and capital intensive machining tools. Chick et al (2000) claim that this model was developed after several interviews with buyers and suppliers of machine tools. The multi-attribute model is shown in figure-2.5.

The multi-attribute emphasize the buyer-supplier relationship that is important for a long-life capital intensive purchase. One of the aims behind of development of this model is to identify the parts of the supplier selection process that can be supported by decision support systems; in this sense, Chick et al (2000) work has some aims that are similar to our work; but our aim is to identify the steps in supplier selection process that can be automated.

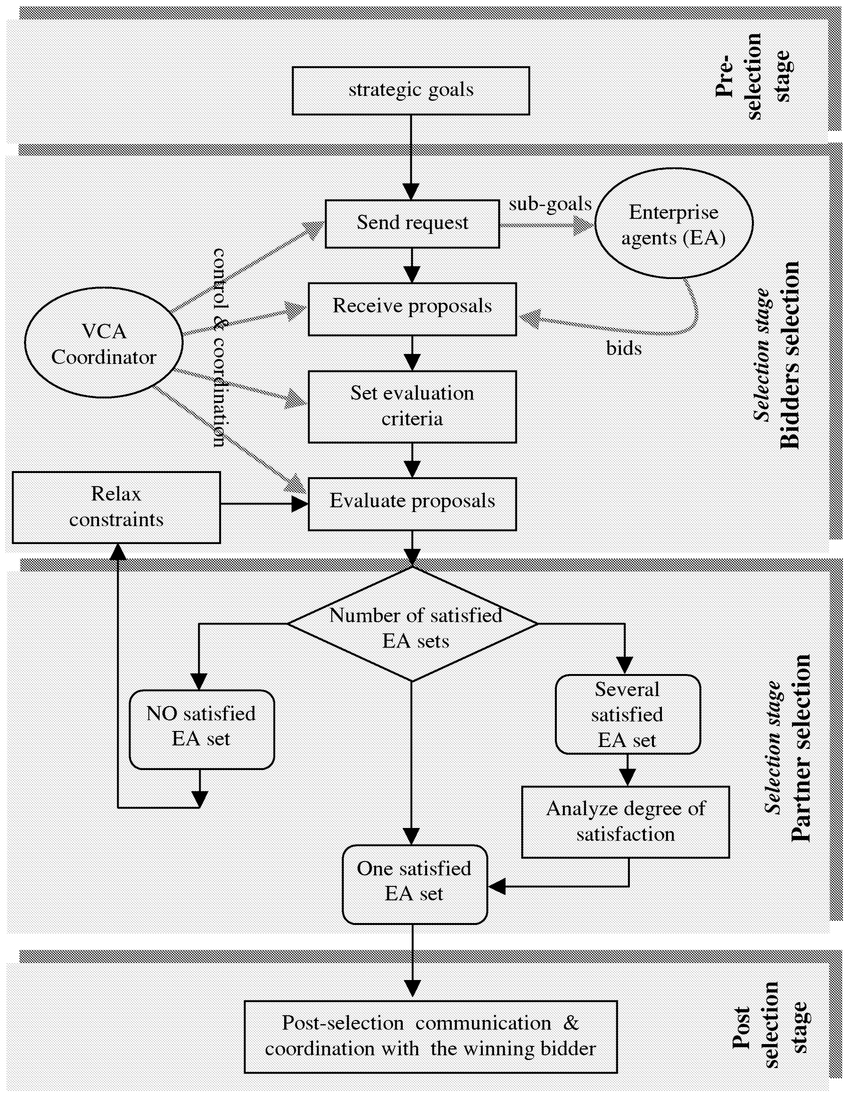
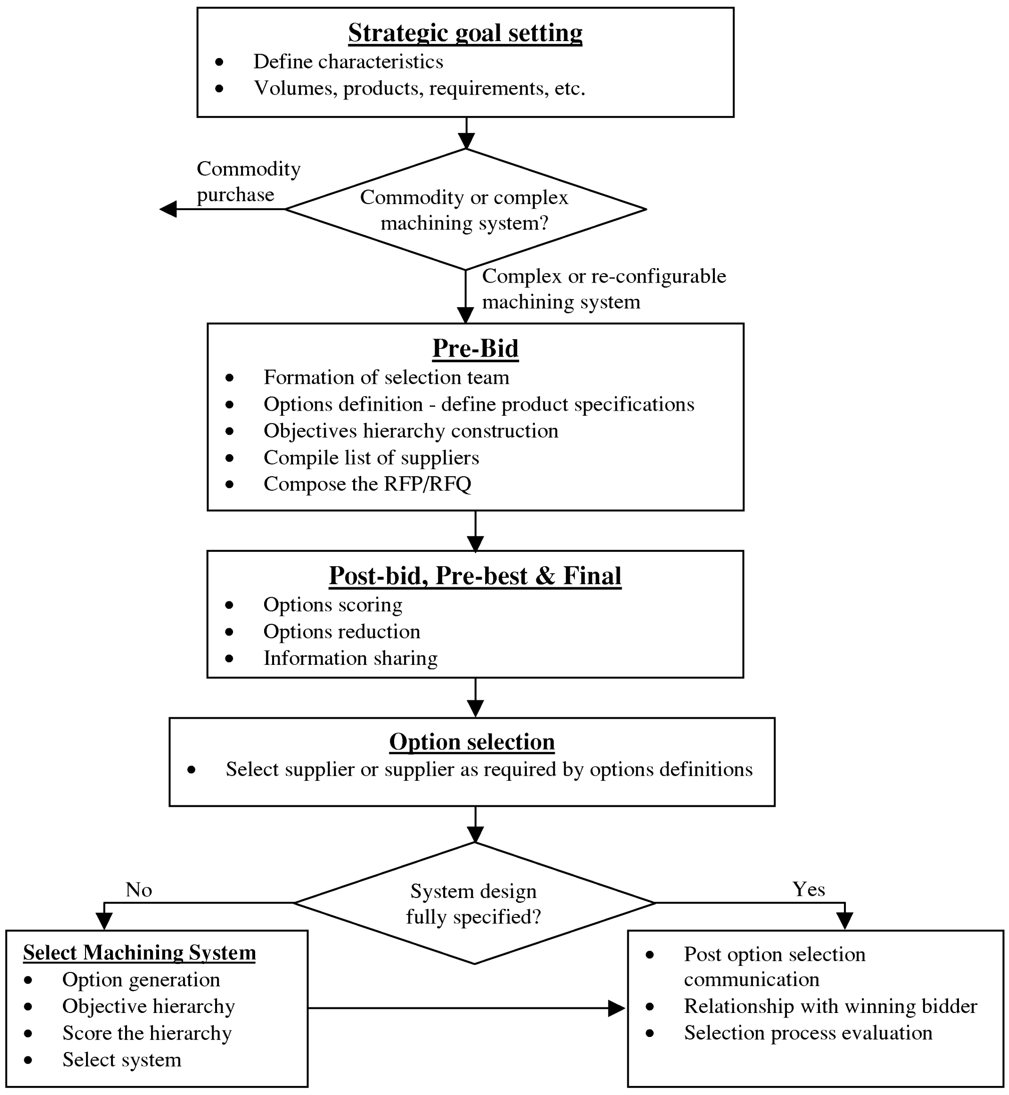


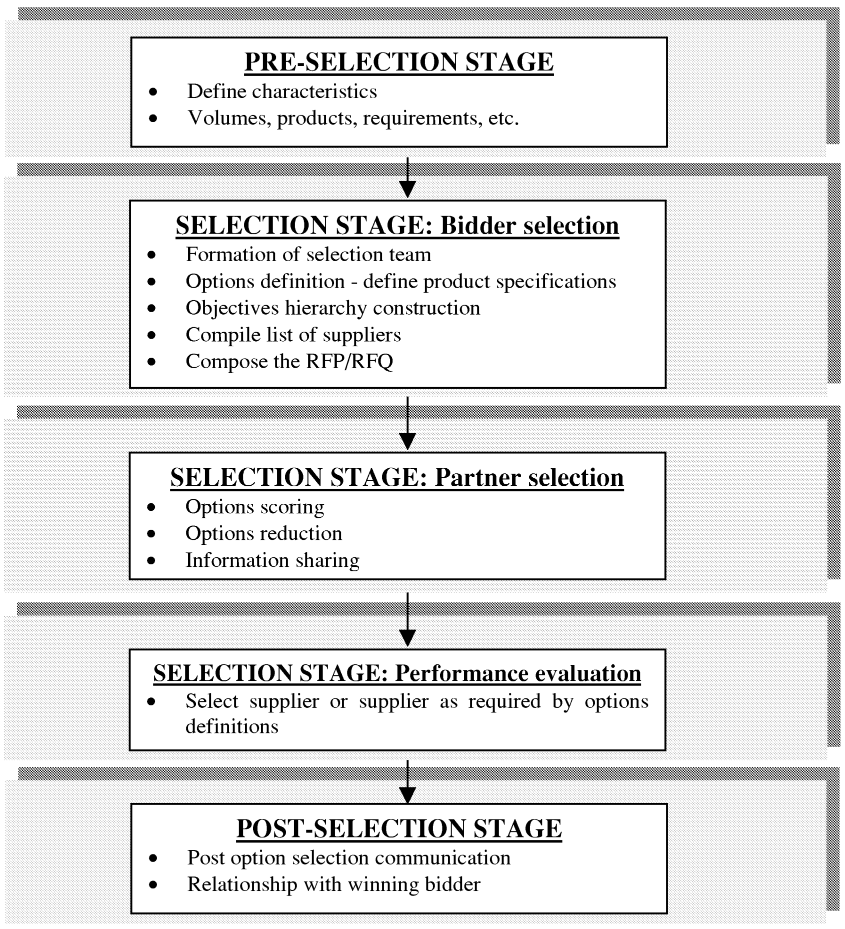
Figure-2.4: Re-modeling 'multi-agent model' by our modeling approach

The revised multi-attribute model created by our modeling approach is shown in figure-2.6. Comparing the two models (shown in fgure-2.5 and 2.6) reveals that though they look similar; the revised model shown in figure-2.6 is a good starting point to explore the possibilities of automating supplier selection procedures. We want to emphasize that in-order to use the methodology we present in chapter-3, the supplier selection procedures must be divided into the three stages, as our methodology automates only the steps that falls within the selection stage.



Strategic goal setting
Define characteristics
Volumes, products, requirements, etc.
Commodity purchase
Commodity or complex machining system?
Complex or re-configurable machining system
Pre-Bid
Formation of selection team
Options definition - define product specifications
Objectives hierarchy construction
Compile list of suppliers
Compose the RFP/RFQ
Post-bid, Pre-best & Final
Options scoring
Options reduction
Information sharing
Option selection
Select supplier or supplier as required by options definitions
No
System design fully specified?
Yes
Select Machining System
Option generation
Objective hierarchy
Score the hierarchy
Select system
Post option selection communication
Relationship with winning bidder
Selection process evaluation

Figure-2.5: The multi-attribute model for capital intensive machine tool procurement



PRE-SELECTION STAGE
Define characteristics
Volumes, products, requirements, etc.
SELECTION STAGE: Bidder selection
Formation of selection team
Options definition - define product specifications
Objectives hierarchy construction
Compile list of suppliers
Compose the RFP/RFQ
SELECTION STAGE: Partner selection
Options scoring
Options reduction
Information sharing
SELECTION STAGE: Performance evaluation
Select supplier or supplier as required by options definitions
POST-SELECTION STAGE
Post option selection communication
Relationship with winning bidder

Figure-2.6: The revised multi-attribute model for capital intensive machine tool procurement

2.3 CASE STUDIES

In the previous section, we used our modeling approach to develop models of supplier selection procedures. The common item in these models is the division of the selection process into the three stages, and further division of the stages into steps. The models we derived in the previous section emphasize different aspects of the supplier selection procedures, like international purchasing (C&N model), coordinating processes with a coordinator agent (multi-agent model), and buying capital equipment (multi-attribute model).

In this section, we shall go through some case studies; the two case studies discussed below, show the multitudes of supplier selection procedures in practice. When we go through each case study, we examine whether the supplier selection procedure practiced by the enterprise can be modeled by our modeling approach; that is, we verify whether our modeling approach is applicable to model the supplier selection procedures of known enterprises.

2.3.1 Case Study-I: Dynamic Instruments, Inc., San Diego

Dynamic Instruments, Inc. (founded 1984) in San Diego-USA, can be classified as an SME with its 220 full-time employees[2]. DI is a leading manufacture of industrial and military voice recording and instrumentation products, supplying to the military sector (e.g. U.S. Navy) and commercial sector (e.g. US Postal service).

Current situation analysis - supplier selection procedures for DI [DI, 2000]
DI does not use an automated supplier selection process. The suppliers have to submit their bids to DI on paper, as DI's web pages does not hosts any forms for suppliers to submit their bids electronically; In making the selection, DI will obtain further specifications from the suppliers. The supplier selection procedure is also simple one, with few criteria (besides some ethical ones like bribing or buying influence through offering valuable personal gifts or entertainment, is not acceptable). The supplier selection process is modeled by our modeling approach and shown in tabular form in table-2.3.

From table-2.3, it is clear that the model of supplier selection procedures of DI look similar to the models we have seen previously, also somewhat simplistic; that is, DI does not practice all the sub-stages of the selection stage as prepared for in our modeling approach; for example, after the initial 1st level selection stage for selection of a pool of potential suppliers, DI jumps to the 3rd level selection stage to select the most preferred supplier. Also, the performance evaluation of the most preferred supplier is not vigorous; rather than using mathematical computations for performance measurement, DI evaluate a supplier based on the subjective issues like previous involvement of the supplier with

[2] the number of full-time employees is yet to be confirmed by DI

DI, location of the supplier (how close to DI), and whether the supplier is a US enterprise.

2.3.2 Case Study-II: Kvæner Oil & Gas, Norway

Kvaerner Oil & Gas Norway (KOG Norway) is a large corporation in the offshore sector [KOG, 2000]. KOG's main business is offering complete installations and services to the oil and gas industry, with specialist competence in project management and execution, sub-sea and process technology and products, and maintenance and operation of platforms. KOG has five divisions (in KOG's terminology, "business streams") which work closely together, both in projects and in the development of new technology and new products.

We modeled supplier selection procedure by our modeling approach; the model is shown in table-2.4. Table-2.4 clearly indicates that the three-stage selection process is further divided into many sub stages, owing to the complexity and cost of procurement.

The supplier selection procedure starts with the pre-selection stage at which the management sets the criteria for procurement such as local contents requirements, logistics, lead time, costs and service possibilities.

stage	*sub-stage*	*steps*
Pre-Selection	(Strategic Goal Setting)	(Corporate) management sets the strategic goals for the purchase. Management also defines the criteria such as local contents requirements, logistics, lead time, costs and service possibilities.
Selection	"Bidders Selection"	The supplier is ISO 9001 compliant (quality factor).
		The price of the product (price factor)
		The good reputation of the supplier for adhering to specifications and delivery schedules (agility factor)
		The dependability and service record of the supplier, and the nature of the guaranty and warranty of the product, and the supplier's adequate financial strength (performance factors)
	"Performance Evaluation"	Preferred supplier is: 1. The suppliers who are already involved with DI, 2. Suppliers located near the company operation, and 3. Suppliers who qualify under the "Buy American Act".
Post Selection	(Selection of the most preferred supplier)	Continuous communication with the supplier on materials, product development, costing, product testing, etc.

Table-2.3: A model of supplier selection procedures of *Dynamic Instruments Inc, US*

In the selection stage, the first sub-stage is the "bidders selection stage", at which the competence of the potential suppliers are measured. The references, quality assurance system, internal control, health, environment and safety policies of the bidders are investigated at this stage. After this stage, a group of qualified suppliers are selected for further scrutiny. The second sub-stage under the selection stage is the "partner selection stage", at which all the request for proposals have been received from the qualified suppliers and under evaluation. At this stage, the relationship or fitness of the supplier with the rest of the manufacturing enterprises is measured.

The final sub-stage of the selection stage is the "project level evaluation" at which the supplier is placed on a specific project situation to evaluate how it will perform in collaboration between the customers, KOG and the rest of the collaborating enterprises.

The final stage, the post selection stage has a number of steps; the main job at this stage is communicating with the winning supplier.

stage	*sub-stage*	*steps*
Pre-Selection	(Strategic Goal Setting)	(Corporate) management sets the strategic goals for the purchase. Management also defines the criteria such as local contents requirements, logistics, lead time, costs and service possibilities.
Selection	"Bidders Selection"	Ability of the supplier to provide support in pre-sales
		Good references from offshore oil and gas industry
		Ability to meet typical delivery times
		Project execution capability, dedicated people, availability of testing facilities.
		Broad product range satisfying ISO standards
	"Partner Selection"	Fit for close relationship
		Performance in example configuration
		Technology partnering capabilities, operation capabilities
	"Performance Evaluation"	Operator and region specific criteria Additional criteria for a Specific Project; on work load, technical configuration, commercial performance
Post Selection	(Selection of the most preferred supplier)	Continuous communication with the supplier; Key suppliers are kept up to date on issues, such as planned projects
		Basic relationship models: one time contract, short term commitment, close relationship, alliance or strategic partnership
		Value added relationship evaluation
		Joint development initiatives

Table-2.4: A model of supplier selection procedures of *KOG, Norway*

2.4 SUMMARY

In this section, a modeling approach is devised for modeling supplier selection procedures. The aim of this approach is to model the selection processes of an enterprise so that the steps of the selection procedures that can be automated are easily identified for automation.

The modeling approach partition the selection procedure into three stages:

- Pre- Selection stage: Management sets the strategic goals for procurement.
- Selection stage: The main selection procedures, starting with many potential suppliers and ending with a most preferred supplier. The selection stage is further divided into three sub-stages called bidder selection, partner selection and performance evaluation.
- Post- Selection stage: Establishing collaboration with the selected supplier.

The modeling approach is shown in tabular form below (table-2.5). By this modeling approach, we were able to model some supplier selection procedures mentioned in literature as well as in industry.

stage	***sub-stage***	***steps***
Pre-selection stage	Strategic goal setting	Management sets the strategic goals for procurement; also defines criteria such as low cost, JIT delivery, high quality etc.
Selection stage	Bidders selection (1st level selection)	Make the request for proposal (RFP)/ request for quotes (RFQ); Receive quotes from suppliers and select a pool of potential suppliers satisfying the basic requirements (such as cost, quality, etc.)
	Partner selection (2nd level selection)	Analyze the supplier quotation and selection best suppliers based on the numerical calculation results
	Performance evaluation (3rd level selection)	The selected supplier is placed in a collaborative environment for a specific project, and performance evaluation is done to see whether the supplier will perform well in collaboration.
Post-selection stage	Selection of the most preferred supplier	Continuous communication with the selected supplier on materials, product development & testing, costing, etc.

Table-2.5: Our basic modeling approach for modeling supplier selection procedures

The pre-selection stage and the post-selection stage are mainly about managerial issues that are subjective. Only the steps that fall within the selection stage are considered further for automation.

3. A METHODOLOGY FOR AUTOMATING SUPPLIER SELECTION PROCEDURES

This chapter is about a methodology for automating supplier selection procedures. We start with our model for supplier selection, the model that was developed in the previous chapter. We then identify the steps in the model that can be automated. And finally show how these steps can be automated.

3.1 AUTOMATING STEPS IN SUPPLIER SELECTION

Table-3.1 shows the modeling approach that was developed in the previous chapter. In addition to the stages, sub-stages, and steps for supplier selection, table-3.1 also shows the steps that can be automated and the modules (*the data collection system*, *the inference engine*, and *the performance evaluation engine*) that perform the automation. Other than the column "automation" under which the modules responsible for automation are given, the only change in the model shown in table-3.1 and the model developed in the previous chapter is the change of the name of the 1st level selection sub-stage to "on-site" selection (from "bidder" selection).

stage	***sub-stage***	***automation***	***steps***
Pre-selection	Strategic goal setting	-	1. Definition of need
		-	2. Review purchase requisition
Selection stage	**On-site selection** (1st level selection)	-	1. Prepare RFQ
		Data collection	2. Source and product identification
		Data collection	3. Locate potential suppliers
		Data collection	4. Accept supplier based on broad margin
	Partner selection (2nd level selection)	Inference engine	1. Re-evaluate qualified supplier list
		Inference engine	2. Analyze supplier quotes
		Inference engine	3. Perform comparison analysis
		Inference engine	4. Rank order suppliers
		Inference engine	5. Prepare final list
	Performance evaluation (3rd level selection)	Performance evaluation	1. Analyze subjective issues
		Performance evaluation	2. Perform final comparison
		Performance evaluation	3. Select supplier
		-	4. Negotiate prices and terms
		-	5. Place orders
Post selection	Relationship maintenance	-	1. Manage the contract

Table-3.1: The model of supplier selection procedure

It is visible from table-3.1 this dissertation only deals with automating supplier selection procedure that falls within the selection stage.

3.1.1 The pre-selection stage

The pre-selection stage ("strategic goal setting") is where the management first takes decision on profitability of a product that is going to be produced, and then sets the strategic goals for the procurement and the criteria for purchase such as local contents requirements, logistics, lead time, costs and service possibilities. Certainly, the management will be using ERP software to assist taking decisions, but this stage can not be fully automated, that is we can not leave the decision making process (like setting goals) to a machine entirely.

3.1.2 The post-selection stage

The post selection stage is at which the management establish relations with the selected supplier; Finishing off the final deals, maintaining a good communication with the suppliers, monitoring, quality control of the received materials and parts from the suppliers, reporting about the difference in the agreed and received supplies, etc. are to be done by the humans.

3.1.3 Automating steps in the selection stage

As shown in table-3.1, many steps within the selection stage can be automated. The selection stage is divided into three sub-stages to help automation. The first sub-stage is naturally about the collection of supplier data; this selection sub-stage is called the on-site selection stage, because the selection is done on the suppliers' web servers. The steps within the on-site selection stage is automated with the help of the data collection system. The second sub-stage of the selection stage is the partner selection stage, where a supplier selected out of many competing suppliers by analyzing suppliers data. The inference engine is used to automate the steps within the partner selection stage. And finally, the performance evaluation engine is used to automate the performance evaluation stage, which is the third sub-stage of the selection stage. At this stage, a selected supplier is checked whether it will perform well during the collaboration. Detailed description about automating these stages are given in the next section, see also table-3.2.

3.2 THE MODULES FOR AUTOMATING SUPPLIER SELECTION

Before forming a virtual enterprise, profitability of a product (that is going to be produced) has to be assessed. This is done in the opportunity identification part of the formation phase. Profitability of a product is assessed by extensive market analysis and research. After the opportunity identification part, the right collaborating enterprises must be found to manufacture and distribute the product; this is the collaborator (supplier and distributor) selection part of the formation phase. Making the selection criteria for supplier selection, and searching for the suppliers, and accepting a supplier as a collaborating enterprise are done in this supplier selection part. The first thing done under this part is the preparation of the supplier selection criteria list (see the model given in table-3.1). Then the main assembler employs three modules to automate the supplier selection; these applications of these three modules are summarized in the following subsections, see also table-3.2. The next three chapters fully describe about these three modules.

Selection stage	*Module responsible*	*Purpose*	*Deciding factors*	*input*	*output*
1. On-site selection	Data Collection System	To select a supplier after reading its product data and supplier quotes. Selection is done on the suppliers' server; if selected, then the supplier data is taken back to the main assembler.	Broad margins for delivery time (agility), cost (leaness), and quality (ISO standard)	supplier web pages in XML	A list of potential suppliers
2. Partner selection	Inference engine	Selecting a supplier out of competing suppliers based on numeric performance. The inference engine uses a mathematical logic model instead of pure mathematical models.	Overall production costs, quality assurance, and response time.	A list of potential suppliers	Selected supplier
3. Performance evaluation	Performance evaluation engine	To evaluate how the selected supplier will perform in collaboration. This collaboration is to produce a specific product / project.	performance in collaborative environment for a specific project	Selected supplier	Accepted partner

Table-3.2: Different sub-stages of the selection stage for automation

3.2.1 On-site selection stage with data collection system

In the on-site selection stage, mobile agents are launched by the main assembler to seek data from supplier web sites. The selection is done by the mobile agents themselves, on suppliers' server. After reading the supplier data, which is encoded in XML format, the mobile agent selects the supplier subject to the broad selection margins set for data like delivery time (agility), cost (leaness), quality (ISO standard), etc.

The main aim of the on-site selection stage is to decide whether a supplier satisfy the *critical performance issues* (such as delivery, cost, quality). To enable the mobile agent to make such decision, it is equipped with a simple logic controller; the mechanism of the logic controller is explained below:

The in-built logic controller of the mobile agents

First we define the *critical performance measure* (CPM) as a multiplicand of three factors, one for each critical performance issues; if there are more than three critical issues (for example, the main assembler could set "buy-British" as a critical performance issue) then CPM will be a multiplicand of many factors, the number of which will be equivalent to the number of critical performance issues. For each critical performance issue, there will be two margins, an upper margin and a lower margin, both of which are set by the assembler. For example,

CF_T: Time factor = (LOWER_DELIVERY < delivery_time <= UPPER_DELIVERY)
CF_C: Cost factor = (LOWER_COST < cost <= UPPER_COST)
CF_Q: Quality factor = (LOWER_Q_INDEX < quality <= UPPER_Q_INDEX)
.....

$CPM = CF_T * CF_C * CF_Q * .....$

The evaluation of each critical factor will result in either 1 (meaning, the critical factor is within the broad margin) or 0 (meaning, the critical factor falls outside the margin), see table-3.3. Since CPM is a multiplicand the of the critical factors, it will also result is either 1 (means selection of the supplier as a potential supplier for further scrutiny) or 0 (means, the supplier is rejected).

Supplier	*critical factors*			***CPM***
	cost	*delivery*	*quality*	
S_1	0	0	0	**0**
S_2	0	0	1	**0**
S_3	0	1	0	**0**
S_4	0	1	1	**0**
S_5	1	0	0	**0**
S_6	1	0	1	**0**
S_7	1	1	0	**0**
$\boldsymbol{S_8}$	**1**	**1**	**1**	**1**

Table-3.3: on-site selection of a supplier based on critical performance measure

If a supplier is selected as a potential supplier, then the supplier data is taken back to the main assembler. Thus, output of this on-site selection stage is a list of potential suppliers.

The simple logic controller enabling the mobile agent to make decisions on supplier data with the help of the critical factors and CPM is similar to the procedures discussed in [Houshyar and Lyth, 1992] and [Brown and Gibson, 1980].

A final note on cost factor: It is important to note that in addition to the supplier quotes, there are many costs that will incur due to international purchase such as export taxes, international transportation cost, insurance & tariff, costs of money, risk of obsolescence and rejects due to transportation, employees travel costs, survey & inspection costs, etc.

A complete list of costs is given in [Carter and Narashiman, 1990]. Ideally, these costs must be considered when calculating critical factor for cost.

The design and implementation of the data collection system for the on-site selection stage is described in chapter 4.

3.2.2 Partner selection stage with inference engine

After initial selection of a supplier by the mobile agent (on-site selection stage), the product data from the supplier is brought back to the main assembler. Among these competing suppliers, the best supplier for the main assembler can be determined by the distribution of the expected performance scores from the numeric performance measures on overall production costs, quality assurance (or adopting to ISO standards), response-time (ability to meet random fluctuations in demand) and flexibility (capability to tailor product). It is possible that by on-site selection, mobile agents bring supplier quotes from numerous potential suppliers. Thus, when realizing the inference engine, the technology that is used to implement the inference engine must offer fast computation to process these numerous data. We make use of array-based logic (a mathematical logic system) for realizing the inference engine.

The design and implementation of the inference engine is described in chapter 5.

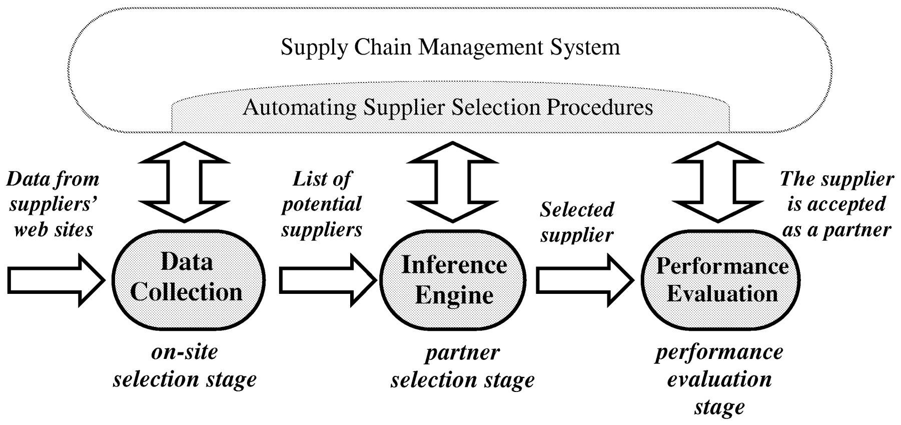
3.2.3 Performance evaluation in collaboration

When a supplier passes the second stage of the selection process (partner selection stage), it has to go through the final selection stage before being accepted as a collaborating enterprise. In this stage, the numeric performance measure of the supplier is extended to include all other already accepted collaborating enterprises to see whether the supplier will perform satisfactorily under collaboration.

The performance evaluation engine does not consider the supplier's potentials for dependable partnership based on its economic strength (financial strength, *'Keiretsu'*- or financial partners, inventory levels), infrastructure (communication lines, country regulations and standards, regional standards, exchange rate implications), and experience (years in business, leadership, goodwill); the top management of the main assembler enterprise should make evaluation of these factors.

The design and implementation of the performance evaluation engine is described in chapter 6.

3.3 SUMMARY



Supply Chain Management System
Automating Supplier Selection Procedures
Data from suppliers' web sites
List of potential suppliers
Selected supplier
The supplier is accepted as a partner
Data Collection
Inference Engine
Performance Evaluation
on-site selection stage
partner selection stage
performance evaluation stage

Figure-3.1: The three modules for automating supplier selection procedures

The methodology for automating supplier selection procedures divides the supplier selection stages into three stages; the pre- selection stage, the selection stage, and the post selection stage. The selection stage is further divided into three sub-stages by the methodology, where each stage employs a module for automation. The three sub-stages are on-site selection stage, partner selection stage, and the performance evaluation stage, see figure-3.1. The modules used for automation at these sub-stages are the data collection system, the inference engine and the performance evaluation engine respectively.

4. DATA COLLECTION SYSTEM

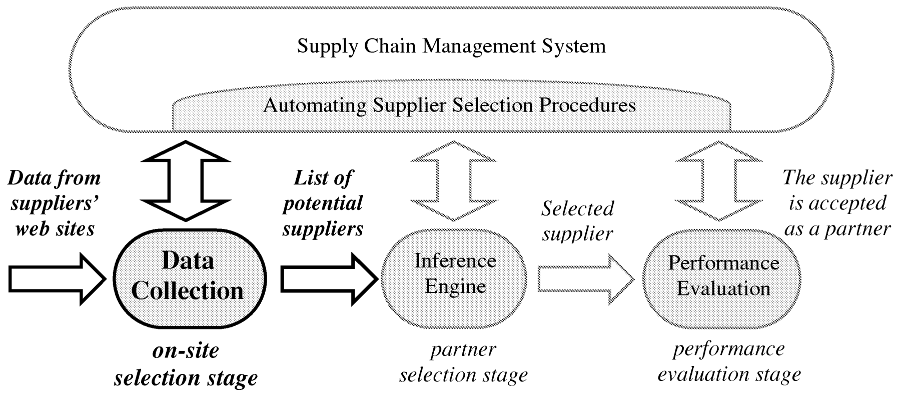

Supply Chain Management System
Automating Supplier Selection Procedures
Data from suppliers' web sites
List of potential suppliers
Selected supplier
The supplier is accepted as a partner
Data Collection
Inference Engine
Performance Evaluation
on-site selection stage
partner selection stage
performance evaluation stage

Figure-4.1: The data collection system as a part of automating supplier selection procedures

This chapter is about data collection system; Data collection is the starting point for automating supplier selection procedures as shown in figure-4.1. The primary function of data collection systems is to collect supplier quotations from suppliers' web sites, and to bring the data to the main assembler's computer system for further analysis. As explained in the previous chapter, the data is accepted from the suppliers' web sites ('on-site' selection), only if the data satisfy preliminary requirements.

Also explained in the previous chapter, that supplier selection is for the formation phase of an agile virtual enterprise. The data collection system we have designed, works for both formation phase and operation phase of the agile virtual enterprise. During the formation phase, there will be hundreds, or even thousands of *potential* suppliers; during the formation phase, the data collection system will perform as an open systems, gathering data from the potential suppliers. During the operation phase, there will be only a specified number of collaborating enterprises; in this case, the data collection system will serve as a closed system, providing information infrastructure to the collaborating enterprises. It must be emphasized that when we talk about formation phase, we are referring to re-configuration phase as well.

During modeling, designing and implementation of the data collection system, the following qualifying factors were taken into consideration; the data collection system should be,

1. *portable*: That is, the data collection system should be able to be installed and run on different platform (different hardware, operating systems) with minimal or no modifications.
2. *interoperable*: That it should be able to use the resources and software running on different platforms.

3. *scalable*: Agile virtual enterprise deals with hundreds of collaborating enterprises and perhaps thousands of potential collaborators; thus the system should be designed to tackle this large number of enterprises, users and transactions.
4. *extensible*: The ability to permit additional functionality over time; specially, the incorporation of legacy software systems.
5. *secure*: That if offers basic security and safety mechanisms. Security and safety measures against tampering and misuse of the data (on transmission or not), and of the computing systems must be guaranteed by adhering to the standards like secure socket layer (SSLv3), encryption, etc.
6. *affordable*: The system should be cheap to build and maintain, as the system is intended for use by SMEs; should be to easy to build and use.
7. *qualitative*: That the systems offer high quality of services that are fast and reliable.

During the initial design stage of the data collection system, a survey of the *enabling technologies* for information infrastructure in the inter-enterprise environment was done. The survey is very important, as there are too many competing technologies available for implementing the information infrastructure. The first section of this chapter is about the survey.

The second section of this chapter is description of two scenarios; the first scenario is about the formation phase and the second scenario is about the operations phase. These scenarios describe what to expect from the data collection system, whereas the survey on enabling technologies pinpoint where to go for implementing the data collection system.

The third sections about *an architecture* for the data collection system. This architecture provides the blueprints, the structural abstractions, and a style that rationalize arrangement and connection of technology *components* to realize the data collection system.

The fourth, fifth and sixth sections describe implementation of a testing prototype of data collection system for formation and operation phases of an agile virtual enterprise.

4.1 SURVEY ON ENABLING TECHNOLOGIES

Building a data collection system satisfying the 7 requirements (or qualifying factors) stated above is not an easy task. In this section, a survey is done on available enabling technologies that are suitable for realizing the data collection system. The technologies such as Object-oriented technology / Java, Common Object Request Broker Architecture (CORBA), mobile agents and extended markup language (XML) are reviewed here.

4.1.1 Object-oriented technology

The choice of object oriented technology is very important for larger and complex systems such as data collection, because the object oriented technology allows independent construction and stepwise refinement of code that can be reused.

Particularly, the object oriented programming language Java offers some unique features that include portability across platforms. Applications written in the Java language is platform independent, that means, these applications developed for a specific platform (say UNIX) will also run on other platforms (say Windows NT, Mac OS) as well. In addition to this property, Java also offers a cleaner and simpler code (than C++) and component model (Beans) [Vogel and Duddy, 1998]. With several layers of security control protection against malicious code, Java is claimed to be one of the most secure language; Java is a strongly typed language, with security control mechanisms such as byte-code verification, "sandbox-model", and digital signature attachment [Flanagan, 1997]. The main shortcoming is that, because Java complied code are interpreted, they are somewhat slower than C++ compiled code.

4.1.2 Common Object Request Broker Architecture (CORBA)

CORBA is based on distributed object oriented technology and is a vendor independent standard developed by the Object Management Group [Object Management Group, 2000]. CORBA also offers many unique features that include access to objects regardless of their location (location transparency). Other features are interfaces defined independently of implementations, access to standard CORBA services and facilities, and access to objects written in other languages [Vogel and Duddy, 1998]. Access to objects written in other languages (e.g. legacy code) is possible, even across networks, with the help of CORBA's interface definition language (IDL). When used together, network transparent CORBA with Java's implementation transparency yields 'distributed objects'. In addition, Transactional Java Beans based on CORBA Object Transaction Service (OTS) offers atomic, consistent, isolated, and durable (ACID) protection to the distributed objects [Orfali, 1997].

4.1.3 Mobile Agents

The agent view provides a level of abstraction at which we construe of computational systems that inter-operate across networks linking people, organizations, and machines on a single virtual platform [Barbuceanu and Fox, 1996]. Agents typically posses characteristics such as autonomous, adaptive (learning), mobile, persistent, goal oriented, collaborative, flexible, and reactive [Sundsted, 1998; Venners, 1997]. Agents decide where they will go and what they will do, and they control their lifetime ('autonomous'). They may receive requests from external sources, such as other agents, but each individual agent decides whether or not to comply with external request [Venners, 1997]. When a new task is delegated by the user, the agent should determine its goal precisely, evaluate how the goal can be reached effectively and perform the necessary actions ('goal-oriented', 'flexible'). An agent should also be capable of learning from past experience ('adaptive'), and should be 'reactive', so that it can sense the current stage of its environment and act accordingly. Agent interacts with other agents to perform its task ('collaborative'). The Agent is 'persistent' because it is a continuously running process and it is 'mobile' because of its ability to transport itself from one machine to another [Franklin and Graesser, 1996].

Mobile agents offer a great deal of advantages over the 'static' agents (or traditional client-server paradigm):

1. Compared to the client-server architecture, mobile agents reduce the network bandwidth by moving a query or transaction from client (or remote assembler) to server (or local assembler), thus the repetitive request/response handshake is eliminated,
2. Agents reduce design risks by allowing decisions about the location of code to be pushed toward the end of the development effort when more is known about how the application will perform,
3. Because the repetitive request/response handshake is eliminated, mobile agent architecture allows applications run on low-reliable or partially disconnected networks [Sundsted, 1998].
4. Immediate reaction to incoming streams of real-time data by an agent that acts as a digital proxy for a human user,
5. Complex or larger calculations can be broken into simpler units and then assigned to agents that are forked from an "agent farm". The agents can perform the calculations on different hosts, and upon completion, the results from the agents can be summarized [Sommers, 1997].

Because of the large amount of data available from the suppliers (web sites), the (geographically) distributed nature of the suppliers, and because of the action needed for continuously collecting data, the mobile agent is a suitable enabling technology for realizing the data collection system.

4.1.4 eXtended Markup Language (XML)

Because of the fact that XML is extensible, a new set of rules (grammar) can be created, agreed upon, and standardized for product data encoding, say *product-data markup language (PDML)* together with a specification for it (data type definition - DTD). However, defining a grammar is a challenging task, requiring specialized skills and comprehensive domain knowledge. A poor definition of grammar may lead to expensive inefficiencies into processing of data, and the enterprises may soon find that the grammar is insufficient for their work [Leventhal et al, 1998]. At present, there is no publicly agreed DTD for product data description.

4.1.5 Concluding remarks

Java (because of its platform neutrality), CORBA (location transparency), XML (data portability), and mobile agents enable building large heterogeneous systems (such as data collection system).

4.2 THE DOMAIN OF DATA COLLECTION SYSTEM APPLICATIONS

This section is about two scenarios; the first scenario is about the formation phase (also reconfiguration phase), and the second scenario is about the operation phase. By going through these two scenarios, we will be able to decide what to expect from the data collection system.

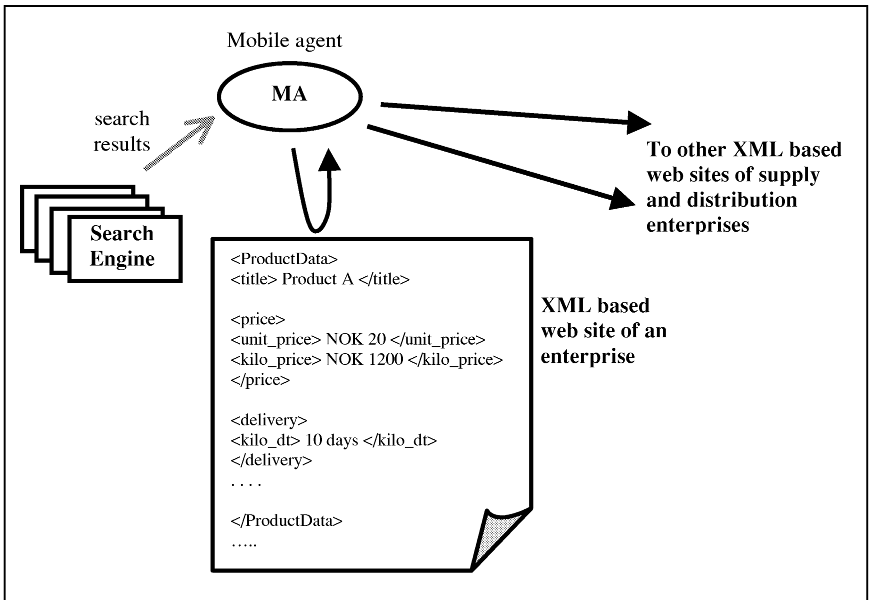
The fundamental difference between data collection in the formation phase and in operation phase is that, in the operation phase, mobile agents are launched only to the collaborating enterprises whose IP address (Internet Protocol address or number; mobile agents visit collaborating enterprises based on this address) are kept in the main assembler's collaborators list (itinerary). Whereas, during the formation phase, mobile agents are sent to any supplier who's web address (URL) is located by the search engine.

4.2.1 Description of the scenario: the formation phase

When the main assembler looks for the potential suppliers and distributors, it looks for data about the price, quantity, delivery time, etc. Traditionally, enterprises turn to outside sources such as professional contacts, trade journals, directories, and import brokers for supplier selection. However, we believe that the enterprises (in our case, the main assembler) can avoid costly and time-consuming middlemen for supplier selection, by using the data collection system for supplier selection that is based on world wide web (WWW) technology, extensible markup language (XML), mobile agent and Java technology. There are many advantages of using a mechanized system over the traditional methods, because (see also [Boyd, 1999; Deadman, 1999; Kojima et al, 1999; Leventhal et al, 1998]):

1. As Internet applications is easy to develop and inexpensive to run, many suppliers will embrace Internet to distribute their information. These suppliers may take information from their databases and render it as XML documents for easy sharing and consumption.
2. A large quantity of data that can be distributed. Though we do not expect that the assembler needs to collect large amounts of data from a supplier web, XML based system does allow a whole document to be transferred if needed.
3. Up-to-date data can be used at all the times. The suppliers can frequently update the web sites with the newest data.
4. In addition to supporting the mobile agents for automatic retrieval of product data, XML based documents are (like HTML) human readable too. Therefore even if the mobile agents based data collection is not utilized, XML documents pave way for human intervention (traditional methods).
5. Java based systems (mobile agents) offer platform neutrality, whereas XML offers data portability. By combining these two technologies, resulting system is suitable for inter-application data exchange that is vital for application sharing among multiple enterprises as in the agile virtual enterprises. A publicly available base DTD (document type definition) may serve as a vehicle for information interchange between enterprises (inter-enterprise), whereas additional DTDs within an enterprise enable separate intra-enterprise interpretation.

The main assembler should launch mobile agents to hundreds of suppliers to fetch these data from their web pages. With the current HTML (hypertext markup language) based web sites, it is not possible to fetch product data, as HTML only deals with the appearance of the web page and does not support automatic retrieval of data by any visiting mobile agents. Another problem is searching potential suppliers' web sites among thousands of web sites. To solve this problem, 1) extensible markup language (XML) and 2) Search engine, could be used as described in figure-4.2 [Davidrajuh and Deng, 2000A].


Mobile agent
MA
search results
Search Engine
To other XML based web sites of supply and distribution enterprises
XML based web site of an enterprise

```
<ProductData>
<title> Product A </title>

<price>
<unit_price> NOK 20 </unit_price>
<kilo_price> NOK 1200 </kilo_price>
</price>

<delivery>
<kilo_dt> 10 days </kilo_dt>
</delivery>
. . . .

</ProductData>
…..
```

Figure-4.2: Data collection using mobile agent and XML

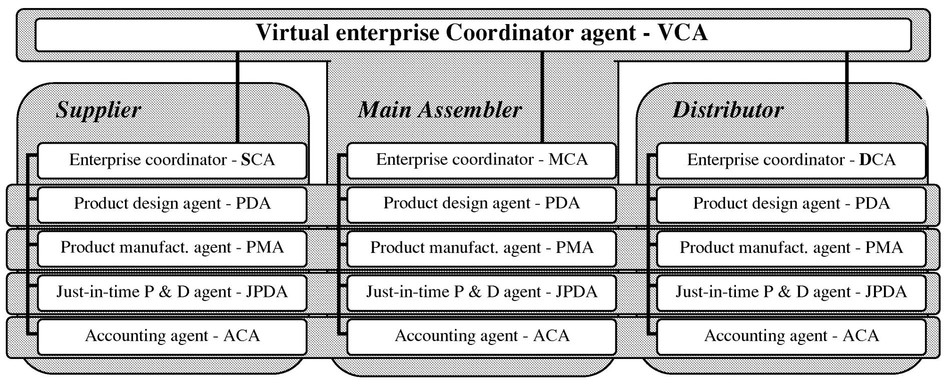
By referencing search engine as shown in figure-4.2, the main assembler gets the web addresses or uniform resource locators (URL) of the web sites of the suppliers that are dealing with manufacturer of a specific product class. Then the main assembler or a unit that is central in forming collaboration- let us call it the main coordinator agent (MCA) sends mobile agents to all the web sites in order to retrieve data from these web sites. To allow this information exchange, the product data on these web pages are tagged with XML tags (for example, <ProductData> <price> … </price> </ProductData>, see figure-4.2) so that the visiting mobile agents can recognize these data. However, before such a page is constructed, there must be agreement on 1) The tags that are allowed, 2) How tags are nested within one another, and 3) How tags should be processed [Bosak, 1999]. There is no agreement for product data description based on XML up to now; until such an agreement is made, our proposed data collection system for formation phase will only be a conceptual one.

After arriving at a supplier's web site, the mobile agent then checks whether the supplier quotations falls within the broad margins set by the main assembler. Only if the supplier data satisfy the broad margins set by the main assembler, then the data is brought back to main assembler for further scrutiny; the mobile agents thus do some selectivity at the supplier web sites, therefore this stage is called the 'on-site' selection stage.

4.2.2 Description of the scenario: the operation phase

During the operation phase, the main assembler coordinates with the supply and distribution enterprises in the supply chain, as shown in figure-1.1. During the operation phase, only a specified number of enterprises participate in the collaboration. Therefore, we need a virtual enterprise coordinator to coordinate with the main assembler coordinator which represents the main assembler and a number of supply enterprise coordinators and distribution enterprise coordinators.

Figure-4.3 shows the coordination and collaboration during the operation phase, with the help of agents [Davidrajuh and Deng, 2000A]; figure-4.3 shows only a single supplier and distributor for brevity. In figure-4.3, a *virtual enterprise coordinator agent* (VCA) coordinates with different enterprise coordinator agents, which are *main assembler coordinator agent* (MCA), *supply enterprise coordinator agents* (SCA), and *distribution enterprise coordinator agents* (DCA).



Virtual enterprise Coordinator agent - VCA
Supplier
Main Assembler
Distributor
Enterprise coordinator - SCA
Enterprise coordinator - MCA
Enterprise coordinator - DCA
Product design agent - PDA
Product manufact. agent - PMA
Just-in-time P & D agent - JPDA
Accounting agent - ACA

Figure-4.3: Vertical and horizontal relationship of agents in a virtual enterprise

The main assembler coordinator agent MCA is agent that requests VCA to fetch data from the other supply and distribution enterprises. These data are then kept in the main assembler local database for its local agents' consumption. Supply or distribution enterprise coordinator agent (SCA or DCA) has tasks such as: 1) Coordinating with VCA to provide the main assembler with new data, to notify change of information, etc. 2) Coordinating internally with its local agents to get the data from them and store it in the local database.

We may define local agents (or called sub-agents) inside an enterprise as *product design agent* (PDA), *product manufacturing agent* (PMA), *just-in-time procurement and distribution agent* (JPDA), and *accounting agent* (ACA), which are under the coordination of enterprise coordination agent as shown in figure-4.3.

In reality, other than *vertically hierarchical relationship* of agents as shown in figure-4.3, there exists also *horizontal relationship* among agents, which reside in different enterprises. For example, when main assembler product design agent PDA is designing products, it may coordinate and negotiate, not only with product manufacturing agent (PMA), just-in-time procurement and distribution agent (JPDA), and accounting agent (ACA) inside the same enterprise, but also with PDAs in supply and distribution enterprises. The *horizontal coordination* among agents is also logically shown in figure-4.3.

4.3 AN ARCHITECTURE FOR DATA COLLECTION SYSTEM

In this section, we present an architecture for data collection system, extending the inter-enterprise information infrastructure of the main assembler to incorporate the information components of the suppliers, main assembler, and distributors. Our main objective is to design a data collection system to realize the scenario described in the previous section; the architecture we present in this section, is to guide ourselves to achieve this objective.

The architecture has the following features:

- It is *component-based*: Component-based application development is an emerging architectural approach where each layer in the architecture offers services to higher layers while hiding the details of how these services are implemented [Fingar et al, 2000].
- It is based on *open standards*: Since there are too many de facto and de jour standards to choose from, it is desirable to adopt open standards, because open standards provides the greatest opportunity to achieve the critical goals of portability and interoperability.

From the scenario description given in the previous section, we clearly identify the following paradigms as the driving-force behind the data collection system:

- *Distributed event* triggering for control and coordination of the whole system ('control activity')
- *Mobile agents* for communication between enterprises ('communication channels')
- *Wrapper agents* tackle semantic mismatches between data sources, legacy sources etc. ('components')

In following subsections, we describe the architecture for data collection system. Also given at the bottom of these subsections, is the implementation techniques, with the help of a prototype.

4.3.1 The three views of the architecture

The architecture presented here starts with a simple component-based overview of the information infrastructure. There are three major actors dealing with the information infrastructure:

1. The information resources and applications,
2. Networks for mobility of information, and
3. Control or supervisory mechanisms for routing and control of information.

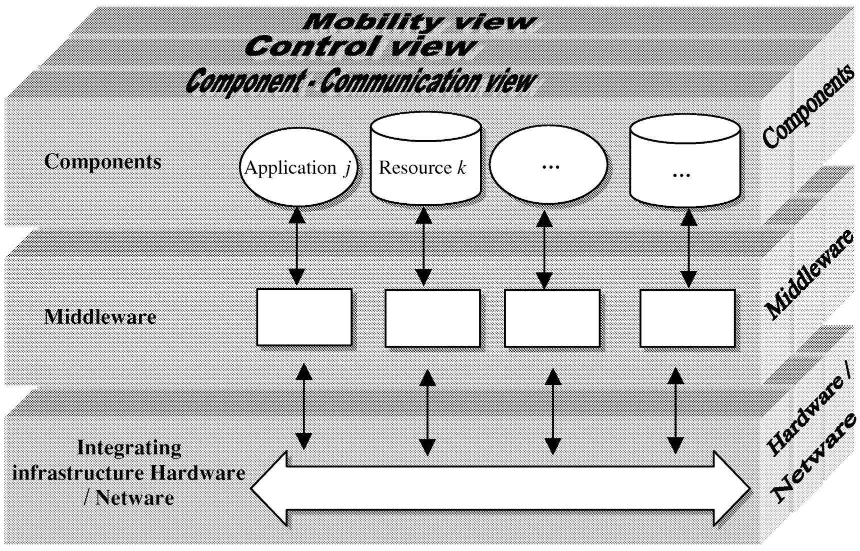


Figure-4.4: Formation of the three views of the architecture

Therefore our architecture too is divided into three inter-related (and overlapping) views (see figure-4.4) [Davidrajuh and Deng, 2000A]. This division is for reducing complexity and allowing independent and parallel development of components under different views.

The three views are 1) Control view, 2) Mobility view, and 3) Component-communication view. Figure-4.4 shows the formation of views of the architecture. Figures-4.5, 4.6 and 4.7 show the detailed exposition of the views [Davidrajuh and Deng, 2000A].

As shown in figure-4.5, in the control view, the middleware is partitioned into two layers of abstraction. The application interface layer represents the connection of the applications and data sources to the integrating infrastructure through agents. In the control layer, there are two central controllers that take basic control actions on agents. These two controllers are the *Agent Manager* and the *Event Manager*.

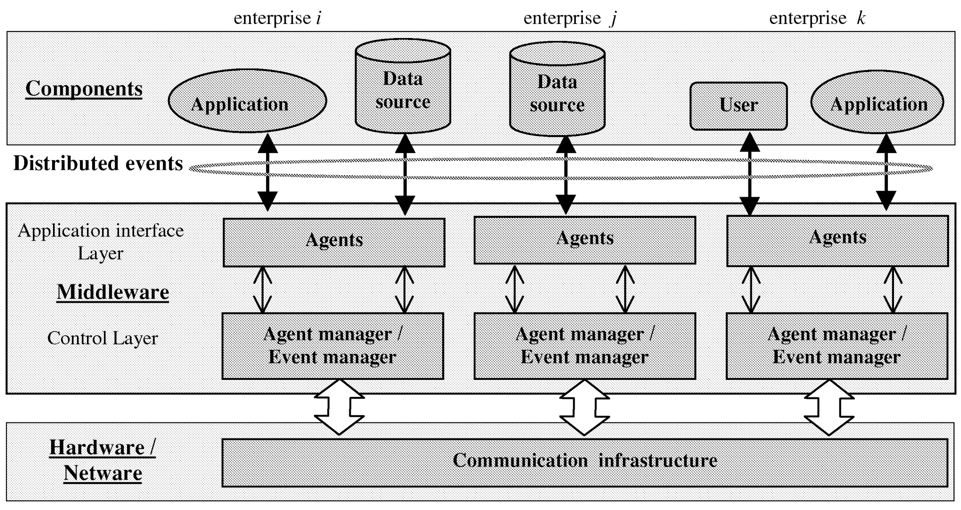


Figure-4.5: The Control view

The agent manager controls the life cycle of the agents and their actions, and offers relevant services relating to this area. Agent startup, execution context, inter-communication, and persistency are under the agent manager control. The event manager controls the distributed event triggering. Event manager registers the agents that wish to be notified about an event. Event manager notifies these agents whenever an event happens for which the agents are registered.

The underlying communication infrastructure refers to the hardware and the net operating system.

From figure-4.5, it is obvious that distributed events paradigm is useful for the situation where applications and data sources are distributed and autonomous. For example, the simpler way for an application to understand the change of state of a data source on the other side is by notification of the change by an event whenever the change occurs.

Figure-4.6 shows the mobility view. In this view multiple resources on heterogeneous platforms are pooled together as a homogeneous system to serve the mobile agents seeking data from these resources to satisfy the applications which started them. To enable this, resources have to be encapsulated, and the request to them and response from them to the agents has to be transparent of where the resources are located. With CORBA technology the kind of common bus for mobile agent transportation as shown in figure-4.6 can be created. To do this, different nodes where diverse applications and resources are attached, should use CORBA object request broker (ORB). Local resources attached to the individual nodes are encapsulated ('wrap-up') by CORBA interface definition language (IDL), and their presence is let known to all the nodes by registering in the local interface repository.

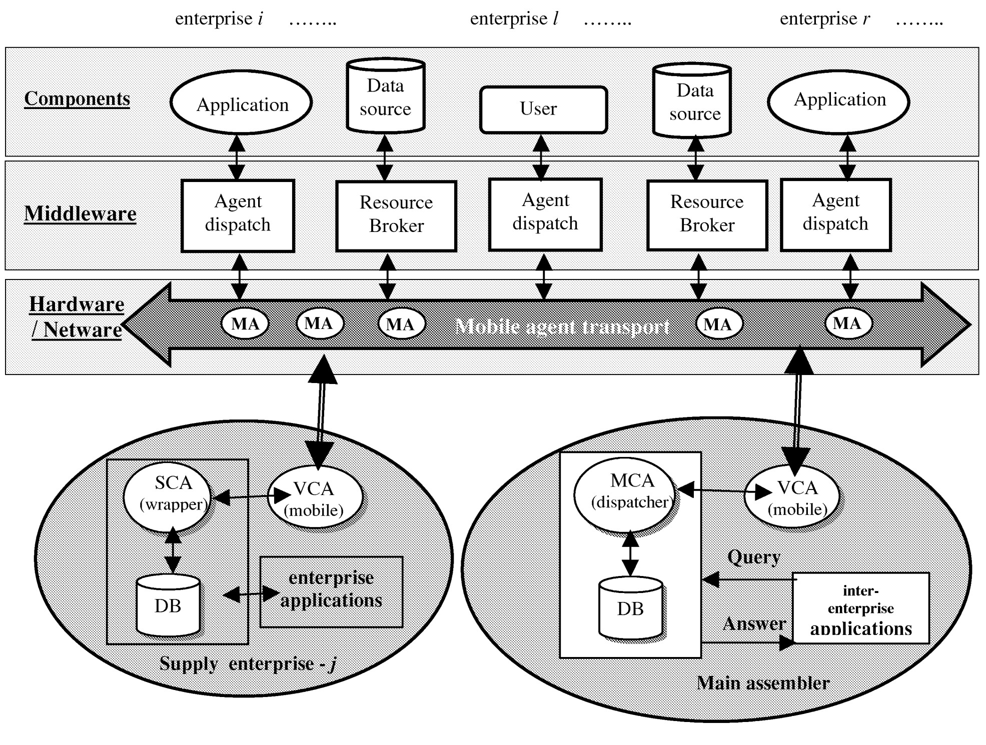


Figure-4.6: The Mobility view

The coordinator agents (MCA, DCA, and SCA) are classified as *wrapper agent* too, providing a level of abstraction between a data source (or simply- database) and requesting mobile agent. The wrapper agent understands how to access the data source and the permission structures associated with it and wrap-up legacy data for transparent cross-platform access. The assembler coordinator agent MCA has an additional function too, which is launching (or dispatching) the mobile agent VCA.

Figure-4.7 shows the component-communication view. This view identifies the components and their communication relationship. There exit several kinds of agents (coordinator agents such as SCA, MCA, and DCA, and functional agents such as PDA, PMA, JPDA, and ACA), and this kind of differentiation of agents strongly influences the architecture. Also, they can be categorized as static agents and mobile agent. In figure-4.7, only VCA is mobile.

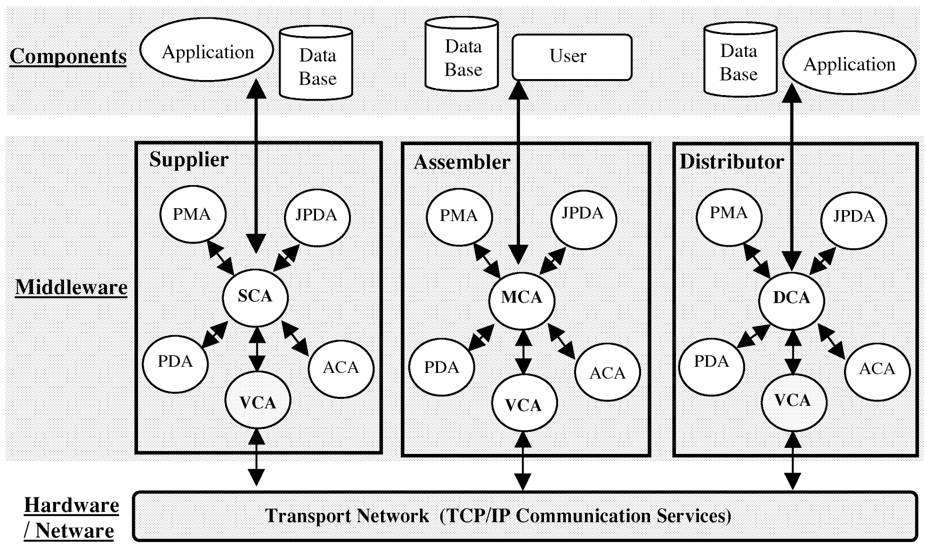


Figure-4.7: The Component-communication view

The component-communication view also identifies the agent inter-communication mechanism and their relevance to the communication infrastructure. For example, what are the groups of agents that are collaborating with each other (in both intra- and inter enterprise), what are the vertical and horizontal relationships that exist between these agents, and how to realize these relationships, have to be identified in this view. In figure-4.7, the inter-enterprise vertically hierarchical relationship among agents is pictured by MCA on the top of the hierarchy by controlling the launching and directing of VCA, and other coordinator agents (SCA/DCA) in a way, acting as slaves. The horizontal relationship between the functional agents (inter-enterprise) could be realized by exchanging data with the help of mobile VCA.

4.4 TOWARDS IMPLEMENTATION OF DATA COLLECTION SYSTEM

In the next two sections (4.5 and 4.6), we present the testing prototypes as demo implementations of data collections system for formation and operation phases; we followed the architecture described in the previous section for implementation. Even though the prototypes are crude, they prove that the ideas presented in this work are realistic.

In this section, we present some remarks regarding implementation of the testing prototypes of the data collection system; first we talk about the limitations in implementing the prototypes, and then we present the enabling technologies used in realization of the prototypes. Also given is the experimental set-up.

4.4.1 Some difficulties in implementing the testing prototype

We faced a number of problems when we designed and implemented a testing prototype of the data collection system for formation and operation phases of an agile virtual enterprise. Some of the difficulties and limitations are discussed here; they are broadly:

1. *Limitations due to different XML standards*: There is no common standard for XML-based applications. Without a common standard, it will be not possible for independent enterprises to agree upon a 'common ground' to exchange data.
2. *Limitations due to non-interoperable mobile agents*: Mobile agents developed by different agent development systems do not talk to each other, that is, not interoperable.
3. *Limitations in programming due to time constraints*: Due to the time limitation compelled upon this Ph.D. research work, the implemented testing prototype is crude, lacking graphic user interfaces (GUIs) and fine-tuning for performance.

Limitations of mobile agent paradigm

One of the main concerns in using mobile agents is the agent server platform, which provides all kinds of services for the agents. There are many agent server platforms (agent development systems such as Aglets [Aglets, 1999], Bee-gent [Beegent, 1999], Concordia [Concordia, 1999], and Voyager [Voyager, 1999]) to choose from. Agents launched by different the server platforms are not inter-operable. This means there is a restriction for errorless operation of the data collection system, that is - it is essential that all the web servers of the many enterprises deploy the same agent server platform.

Limitations of XML-based systems

XML applications do not interact with each other if the tags and data type definitions (DTDs) are different. With the rapid adoption of XML as the meta data format for electronic commerce applications, many XML-based industrial standards (such as BizTalk, CBL, cXML, IOTP, OAGIS, OCF, RETML, the foremost being ebXML and UDDI) have been proposed [Li, 2000; Morgenthal, 2000]. The question of adopting to which standard is the biggest question of every developer face.

For our prototype, we defined new tags to organized the tags into a new DTD. This DTD is very simple, shown later (in figure-4.10, together with a sample XML document based on this DTD). By using this simple DTD, we build the testing prototype. Of course this DTD can not be used for any real-life data collection system, but our aim here is just to prove that our ideas can indeed lead to realization of data collection system for industrial use.

Limitations on programming
The architecture proposed in this paper is based on the integration among small and medium sized enterprises. For this integration, the vertical structure of the agents needs to be extended to posses more levels of hierarchy (referring to figures 4.3 and 4.7); but for the prototype, we shall limit ourselves to the much simpler agent models shown in figures 4.3 and 4.7.

Because of great working load in agent programming, we have only partially implemented the coordinated agents in our testing prototype. We have also ignored development of graphical user interfaces (GUIs), again due to lack of time; all the test results are shown on the standard DOS screen, rather than using Java Abstract Windowing Toolkit (AWT) / (Swing in version 2).

4.4.2 Enabling technologies for implementing the testing prototype

The testing prototype is explained in the next two subsections. In this subsection, we list the enabling technologies used to build the prototype. They are:

1. *The Concordia agent development system (version 1.1):* There are many Java based agent development systems available. We found out that with the Concordia agent development system, implementing the data collection system will conform to the architecture we devised in the section-4.3 [Concordia, 1998; Walsh, 1998]. Also, Concordia offers simple programming interface for distributed event management and for agent mobility. Hence, Concordia was selected as the agent development and management system for our prototype. We considered only the Java based systems because of the well-known pro- Java reasons like platform independency, simplicity and security [Flanagan, 1997; Fuggetta, 1998]; the mobile agent frameworks that are based on other implementation languages (like AgentTcl, which is based on Tcl) were not considered.
2. *The InstantDB Java Relational Database Management System (version 2):* We use InstantDB Java database as the local databases; this is to avoid JDBC-ODBC bridging which is needed if in case other databases (Access, for example) are used. InstantDB is a pure Java database [InstantDB, 1999]. As InstantDB has no built in network support, some RMI based JDBC driver like RmiJdbc [RmiJdbc, 1999] is also needed.
3. *XML for Java (version 1.1.9)*: This is the XML processor used to parse XML documents to construct Java object tree and vice versa. Developed at the IBM's Tokyo Research Laboratory, this parser is available at [XML4J, 1999; Maruyama et al, 1999].
4. *Java Development Kit (JDK) version 1.1.7*: The prototype was tested using JDK v1.1.8 running on Windows NT 4.0 with Service Pack 6. Due to upward compatibility, it should (hence programming codes shown in this chapter) run errorless with newer JDK versions (say version 2.0). The testing prototype should

also run on any other platform due to Java platform neutrality, provided that the platform has a Java Virtual Machine (JVM).

In the following two subsections, the design and implementation of testing prototype is explained through Java programming code. The reader is expected to have some experience in Java programming, as introductory materials to the enabling technologies used are not presented here. The reader is advised to refer the respective user manuals/ programmers manual / and reference manuals for further details.

4.4.3 The experimental set-up

Hardware of the prototype is shown in figure-4.8, consisting of just two PCs, one representing the main assembler and the other representing a supplier. The main assembler was assigned IP address (Internet protocol address) 158.39.25.86 with host name "fagpc086.hin.no". The supplier was given IP address 158.39.25.83 with host name "fagpc083.hin.no". It must be noted that data collection system (and the prototype for it) operates only with the IP addresses, so it does not matter whether the workstations are kept side by side in the laboratory, or they are geographically separated by 12:00 hour time difference. It must be also emphasized that in addition to the main assembler, the collaborating partners can be increased to any number (from one, as it is in the prototype). This can be done by simply notifying the IP address of the 'new' partner to the main assembler, so that this address is written down in the main assembler's itinerary.

With the (physical) experimental set-up in mind, let us go through the programming details, starting with the physical processes of collecting new data when a supplier notifies the main assembler about the availability of the new data.

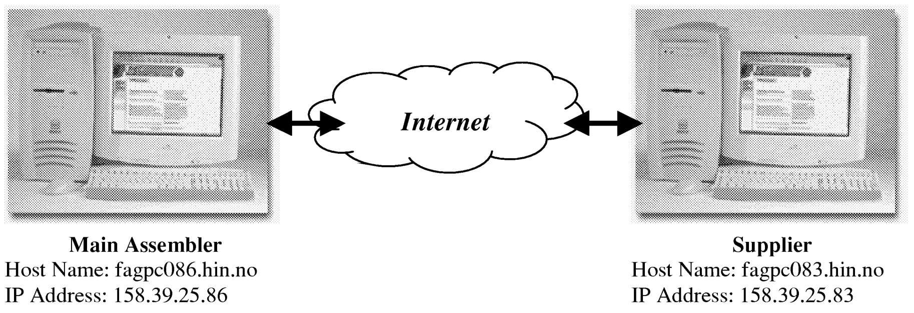


Figure-4.8: The Hardware of the testing prototype

4.5 THE TESTING PROTOTYPE FOR THE FORMATION PHASE

Figure-4.9 shows the components of the testing prototype of data collection systems during the formation phase. The system starts by sending a search request to the search engine. The search request contains the key data about the product needed from the supplier. Upon reception of the search results from the search engine, the agent launcher of the main assembler site launches mobile agents to the supply enterprises identified in the search results.

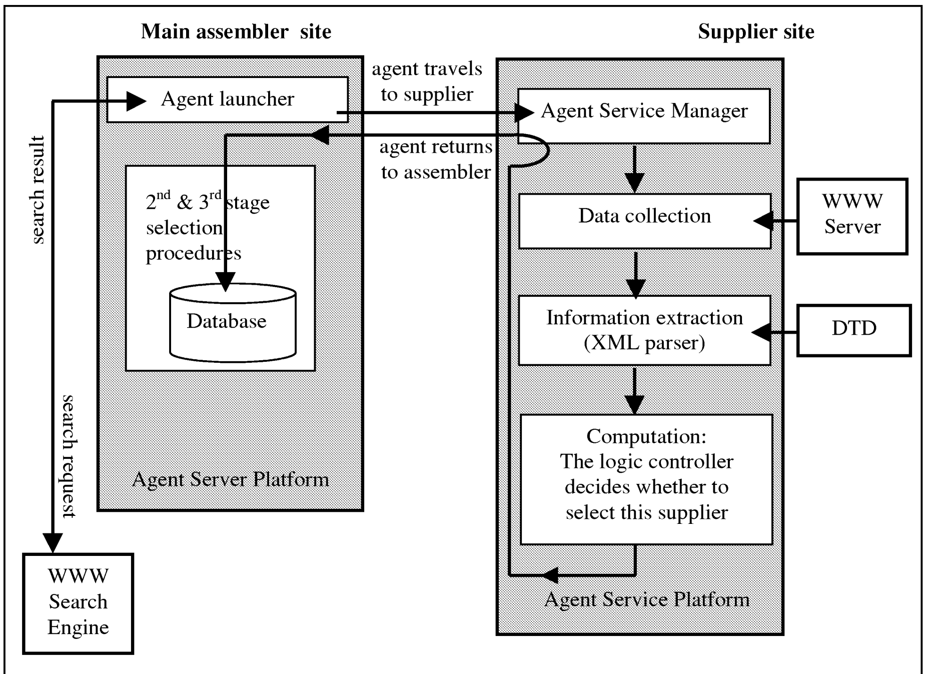

Main assembler site
Supplier site
Agent launcher
agent travels to supplier
Agent Service Manager
agent returns to assembler
search result
2nd & 3rd stage selection procedures
Database
Data collection
WWW Server
Information extraction (XML parser)
DTD
Computation: The logic controller decides whether to select this supplier
search request
Agent Server Platform
WWW Search Engine
Agent Service Platform

Figure-4.9: Data collection during the formation phase

The mobile agent travels to the supply enterprise and collects the product data encoded in XML format. With the XML parser (together with the DTD), this document is examined for key product data such as price, delivery time, and quality. Then the logic controller of the mobile agent determines whether to select this supplier by verifying that the values for the key product data are within the broad margins set for it. If so, the mobile agent takes the XML document with it when it returns to the assembler. Thus, we divide the processes into 5 stages:

1. Using the search engine to locate the supplier URL and XML document
2. Launching mobile agents to the supplier identified in the search result
3. Mobile agent collect data from the supplier on supplier site
4. Mobile agent process the data collected on the supplier site

5. Mobile agent return to the main assembler site, with the data collected if he supplier is selected (supplier quotes are satisfactory).

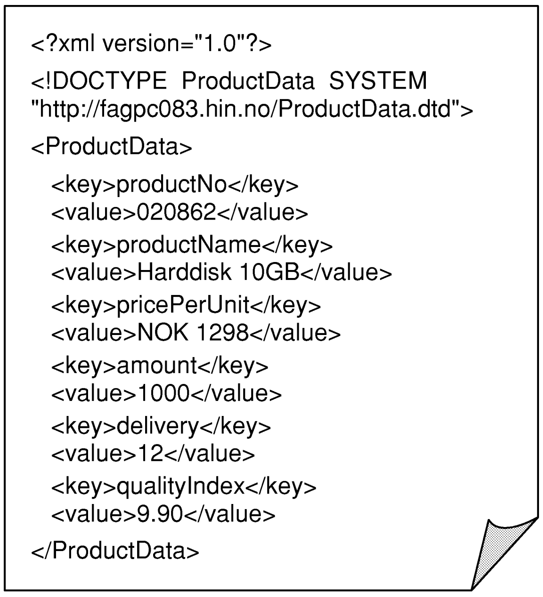
Let us inspect the programming skeleton; note that only the relevant code are shown.

4.5.1 Using the search engine

We have not implemented the module for search engine. We acknowledge that there are some commercial packages available, with which we could use API (application programming interfaces) to manipulate the search engines. However, we believe that this kind of programming comes under computer science / software engineering, thus out-of scope of our interest. Interested reader is referred to [Huck et al, 1998].

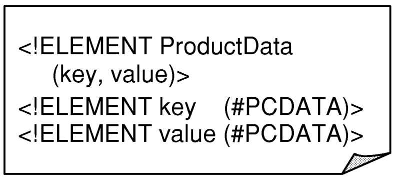
We start with the assumption that the potential the supplier is already located by the search engine, the supplier URL is "fagpc083.hin.no", the XML document is "ProductData.xml", and the DTD is "ProductData.dtd". Sample ProductData.xml document together with the DTD is shown in figure-4.10.

Since the DTD shown in figure-4.10 is so simple, it is not necessary to use a validating parser.



```
<?xml version="1.0"?>
<!DOCTYPE ProductData SYSTEM
"http://fagpc083.hin.no/ProductData.dtd">
<ProductData>
  <key>productNo</key>
  <value>020862</value>
  <key>productName</key>
  <value>Harddisk 10GB</value>
  <key>pricePerUnit</key>
  <value>NOK 1298</value>
  <key>amount</key>
  <value>1000</value>
  <key>delivery</key>
  <value>12</value>
  <key>qualityIndex</key>
  <value>9.90</value>
</ProductData>
```

10a: Sample ProductData.xml document



```
<!ELEMENT ProductData
    (key, value)>
<!ELEMENT key    (#PCDATA)>
<!ELEMENT value (#PCDATA)>
```

10b: ProductData.dtd

Figure-4.10: Sample ProductData.xml document and the ProductData.dtd

4.5.2 Classes and Methods used in the operation phase

Figure-4.11 shows the classes and methods used in the software for implementation of prototype for formation phase; classes are shown using rectangles with rounded corners, and methods are shown using rectangles with square corners. Figure-4.11 also shows where the classes are methods are active (on main assembler site or on supplier site).

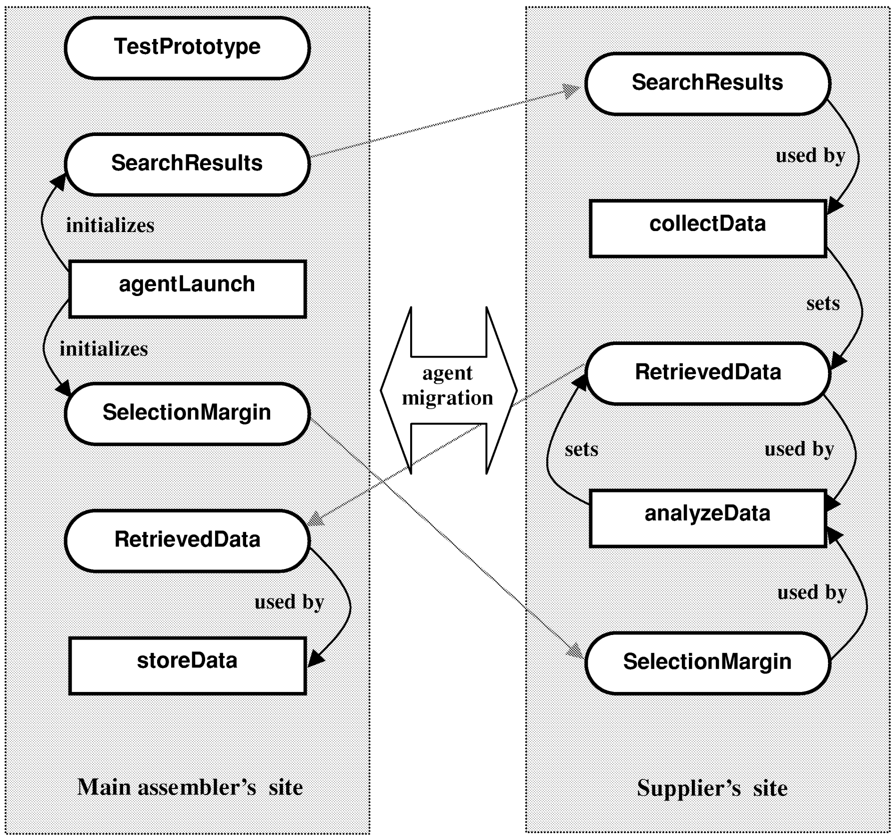


Figure-4.11: Classes and methods used for implementing the prototype for formation phase

Operations on the main assembler site

Class *TestPrototype* is the main class, consisting of all the methods (*agentLaunch*, *collectData*, *analyzeData*, and *storeData*). Method *agentLaunch* does the most of the initialization work, stuffing the search results (supplier URL, location of XML document) into an instance of the class *SearchResults*. *agentLaunch* also sets the margins for selection (maximum allowable price, latest delivery etc.) in an instance of the class *SelectionMargin*. The instances of the class *SelectionMargin* and *SearchResults* will be taken to the destination when the mobile agent migrates. An un-initialized instance of the class *RetrievedData* will be also taken to the destination, where it will be filled-up with the retrieved data.

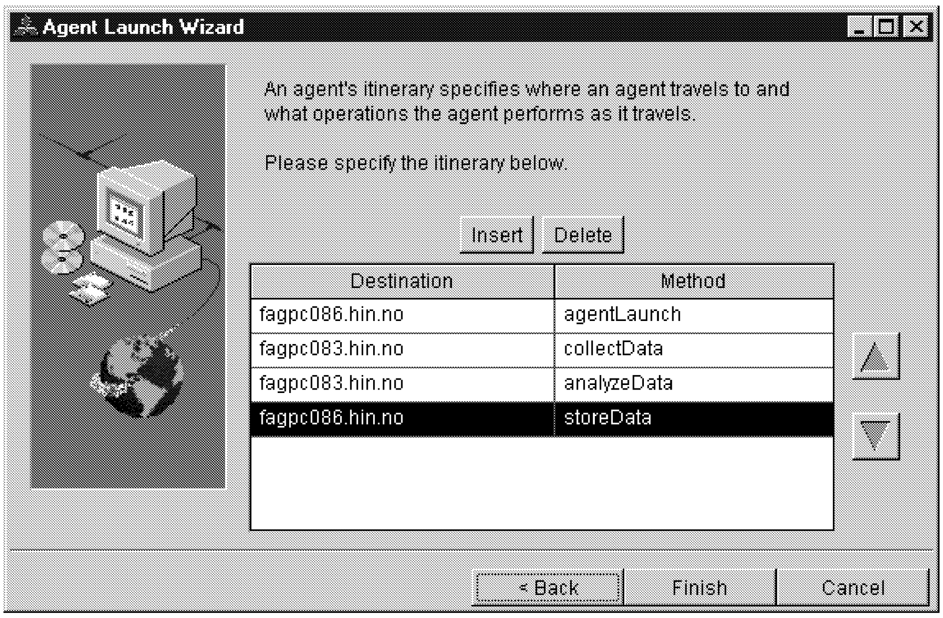
Operations on the supplier site
The method *collectData* will parse the XML document (locating the document with the help of the search results stored in *SearchResults*) then put the retrieved data in an instance of the class *RetrievedData*. Then the method *analyzeData* will analyze the data available in *RetrievedData* using the margins in *SelectionMargin*. If the supplier quotes are satisfactory, then *analyzeData* will also set a Boolean variable in *RetrievedData*. Finally, *RetrievedData* migrates back to main assembler site for reporting.

Back on main assembler site
The method *storeData* will store the data (supplier quote) in main assembler's local Java database if the supplier is selected (Boolean variable is set).

Thus, the classes *SearchResults, SelectionMargin* and *RetrievedData* are used only for the purpose of transporting data (selection margins, search results, and retrieved data, respectively). Only the class *TestPrototype* consists of some methods or executable statements.

4.5.3 Launching mobile agents from main assembler to the potential

After a supplier site is located, a mobile agent must be dispatched. This mobile agent



An agent's itinerary specifies where an agent travels to and what operations the agent performs as it travels.

Please specify the itinerary below.

Destination	Method
fagpc086.hin.no	agentLaunch
fagpc083.hin.no	collectData
fagpc083.hin.no	analyzeData
fagpc086.hin.no	storeData

Figure-4.12: Launching mobile agent from the main assembler to the supplier

takes along two methods with it, one is to perform data collection (that is parsing the XML document) at the supplier web site, and the other method is to analyze the collected data. The method used for data collection is "*collectData*" and the method used for analysis is "*analyzeData*". When the mobile agent returns to the main assembler site, it will store the results is the main assembler database on successful data collection. The method used for storing is "*storeData*".

Figure-4.12 shows the mobile agent destination and the methods to perform at the respective destination. Note that in this figure, "facpc086.hin.no" is the URL of the main assembler and "facpc083.hin.no" refers to the supplier. Figure-4.12 shows the use of Concordia's mobile agent launcher; agent launching can also be embedded in a Java application so that agent launching can be done automatically. To do this an instance of lightweight agent transporter object *COM.meitca.concordia.AgentTransporter* should be created.

The following code shows classes *SearchResults*, *SelectionMargin*, and *RetrievedData*.

```
/** Class SearchResults: consists of the search results obtained
 *   from the search engine  */
import java.io.*;

public class SearchResults implements Serializable {
      public String     locationProductDataXMLDocument;
      public String     locationDatabase;
} //class SearchResults
```

```
/** Class SelectionMargin: consists of selection limits; the limits
 *  are set by agentLaunch. the limits are used by analyzeData.*/

import java.io.*;

public class SelectionMargin implements Serializable {
      public  double    maxPricePerUnit;
      public  int       minAmount;
      public  int       maxAmount;
      public  int       earlyDelivery;
      public  int       lateDelivery;
      public  double    minQualityIndex;
} //class SelectionMargin
```

```
/** Class RetrievedData: consists of data that will be filled by
 *  the method collectdata. The boolean variable supplierSelected
 *  will be set by the method analyzeData. */

import java.io.*;
```

```
public class RetrievedData implements Serializable {
      public  long      productNo;
      public  String    productName;
      public  double    pricePerUnit;
      public  int       amount;
      public  int       delivery;
      public  double    qualityIndex;

      public  boolean   supplierSelected;

} //class RetrievedData
```

The main parts of class *TestPrototype*, showing the codes for the method *agentLaunch* is given below:

```
import java.util.*;
import java.net.*;
import java.util.Hashtable;
import java.io.FileInputStream;
import java.io.DataInputStream;
import java.util.StringTokenizer;
import java.io.InputStream;
import java.util.Hashtable;

import java.sql.*;
import jdbc.idbDriver;
import jdbc.idbResultsSet;

import com.ibm.xml.parser.Parser;
import org.w3c.dom.*;
…
import COM.meitca.concordia.*;
import COM.meitca.concordia.AgentTransporter;
…


/** The main class TestPrototype: consists of all the methods used */
public class TestPrototype extends Agent {

RetrievedData ProductInfo = new RetrievedData();
SelectionMargin selectConstraints = new SelectionMargin();
SearchResults locationStrings = new SearchResults();

public void agentLaunch() {
     try {
        //setting selection margins
        selectConstraints.maxPrice=1200;   //some sample values!
        selectConstraints.lateDelivery=30; //some sample values!
        …
        …

        //stuffing serach results
        locationStrings.locationProductDataXMLDocument =
                                          "ProductDataXX.xml";
        locationStrings.locationDatabase= "//idb191/myDB/myDB.prp";
    } catch (Exception error) {
        System.out.println("TestPrototype.agentLaunch:" +
              " Error occurred\n" + error.getMessage());
        error.printStackTrace();
    } //try
} //agentLaunch()
```

4.5.4 Data collection at the supplier site

Once the mobile agent arrives at the destination (the supplier), it will parse the XML document. The method that is responsible to do parsing is "*collectData*". Given below is the implementation of *collectData*, mainly the programming codes that are relevant to parsing.

```
// THIS METHOD IS ADAPTED FROM [Maruyama et al, 1999]
public void collectData(){

    try {
        //Open XML document
       InputStream is = new FileInputStream(
              locationStrings.locationProductDataXMLDocument);


       //parsing
       Parser parser = new
           Parser(locationStrings.locationProductDataXMLDocument);
       Document doc = parser.readStream(is);

       //exist if parsing error
       if (parser.getNumberOfErrors() > 0) {
            System.exit(1);
       }

       //Creating hashtable for string key-value pairs
       Hashtable hash = new Hashtable();

       String key = null, value = null;

       //Traversing all children of the root element
       for (Node kvchild = doc.getDocumentElement().getFirstChild();
            kvchild != null;     kvchild = kvchild.getNextSibling()) {

           //When child is an element
           if (kvchild instanceof Element) {
               //If tag name is "key", store its content in vkey
               if (kvchild.getNodeName().equals("key")) {
                        key = makeChildrenText(kvchild);

               //If tag name is "value"
               } else if (kvchild.getNodeName().equals("value")) {

                //Extract the text content from the child
                value = makeChildrenText(kvchild);
               //Check key is specified and
               //store the key-value pair int the hashtable
              if (key != null) {
                     hash.put(key, value);
                     key = null;
              }//if (key!=
           }//if (kvchild.getNodeName()
        }// if (kvchild instance
      }//for (Node
```

The above code is not complete; we left it when the hash-table *'hash'* has all the key-value pairs. These pairs are now stuffed into the object *ProductInfo* so that *analyzeData* can utilize the retrieved information. Stuffing key-value pairs into *ProductInfo* is done as shown below:

```

     //Now that hashtable has all the key-value pair, let's stuff
     //it into ProductInfo, so that analyzeData can use it

     try {
        ProductInfo.productNo = new
              Interger((String)hash.get("productNo")).intValue();

        ProductInfo.productName = (String)hash.get("productName");

        ProductInfo.pricePerUnit= new
            Double((String)hash.get("pricePerUnit")).doubleValue();

        ProductInfo.amount = new
              Integer((String)hash.get("amount")).intValue();

        ProductInfo.delivery = new
              Integer((String)hash.get("delivery")).intValue();

        ProductInfo.qualityIndex = new
              Double((String)hash.get("qualityIndex")).doubleValue();

       } catch (Exception error) {
         System.out.println("Error occurred in collectData()\n" +
                                 error.getMessage());
         error.printStackTrace();
     } //try
  } catch (Exception e) {
    e.printStackTrace();
  } //try

} //collectData()
```

4.5.5 Analyzing the data collected at the supplier site

Analyzing the retrieved data to see whether the supplier can be selected is fairly simple:

```
/** method analyzeData: Uses maxPrice, lateDelivery, minQuality etc.
 *  as selection margins to analyze the data in RetrievedData.
 *  If supplier is selected then boolenan variable selectSupplier
 *  in RetrievedData  is set. */

public void analyzeData() {

   //local boolean variables
   boolean priceOkay=false, deliveryOkay=false;
   boolean amounOkay=false, qualityOkay =false;

   try {
      if (ProductInfo.price < selectConstraints.maxPrice)
         priceOkay=true; //Price is okay !!!
      if (ProductInfo.delivery < selectConstraints.lateDelivery)
         deliveryOkay = true; //Delivery is okay
      …
      …
      if (priceOkay && deliveryOkay && amountOkay && qualityOkay)
         ProductInfo.supplierSelected = true; //accept supplier
    } catch (Exception error) {
         System.out.println("TestPrototype.collectData: " +
              " error occurred\n" + error.getMessage());
         error.printStackTrace();
    } //try
} //analyzeData()
```

4.5.6 Mobile agents return to the main assembler site

Here is the final piece of code for storing the retrieved data into main assembler's database, if the supplier visited is selected (Boolean variable *selectSupplier* is set).

```
/** storeData: if the supplier is selected (selectSupplier==true),
 *  then call the method storeDataInAssemblerDatabase for actual
 *  storing. */

public void storeData() {
  String[] SQLstr = new String[100];
  SQLstr[0] = "DROP TABLE tester";
  SQLstr[1] = "CREATE TABLE tester (Prod_No int PRIMARY KEY,
      Prod_Name char(30),  Delivery int, Amount int, Price_PU double)";

  SQLstr[2] = "INSERT INTO tester VALUES (productNo,
        productName, delivery, amount, pricePerUnit, qualityIndex)";
  SQLstr[3] = ".";

  String url   = "jdbc:idb:c://jdb191//myDB//myDB1.prp";

  if (selectSupplier) {
  try {
      idbDriver idb1=new idbDriver();
      Class.forName ("jdbc.idbDriver").newInstance();

      byte[] inBytes = new byte[512];
      Connection con = DriverManager.getConnection (url);
      DatabaseMetaData dma = con.getMetaData ();

      //printing about the database version, driver etc.
      System.out.println("\nConnected to " + dma.getURL());
      System.out.println("Driver    " + dma.getDriverName());
      System.out.println("Version   " + dma.getDriverVersion());
      System.out.println("");

      //Create a Statement object to submit SQL statements to driver
      Statement stmt = con.createStatement ();

      //executing the SQL statement, to store data
      int i=0;
      while (SQLstr[i] != ".") {
         try {
            if (stmt.execute(SQLstr[i])) {  //Result set available
              ResultSet rs = stmt.getResultSet();
              rs.close();
            }//if
             } catch (SQLException e) {
            e.printStackTrace ();
             }//try-catch
            i++;
        }//while

      stmt.close(); //Close the statement
      con.close();  //Close the connection
   } catch (Exception ex) {
      ex.printStackTrace ();
   }//try-catch
  }//if
 }// storeData()
```

4.5.7 Some performance improvement techniques

There are better ways to implement the prototype; For example, in the *collectData* method (subsection 4.5.4), we used Document Object Model (DOM) API for parsing. Since we were only interested in extracting a few information (like price, delivery, quality) from the XML document, we could have used either Simple API for XML (SAX) or ElementHandler API for parsing. SAX and ElementHandler do not create the entire document structure in memory (which DOM API does), therefore faster and efficient than using DOM. These kind small performance improvements were not done in the prototype, because our aim was to build a simple working prototype in the first place.

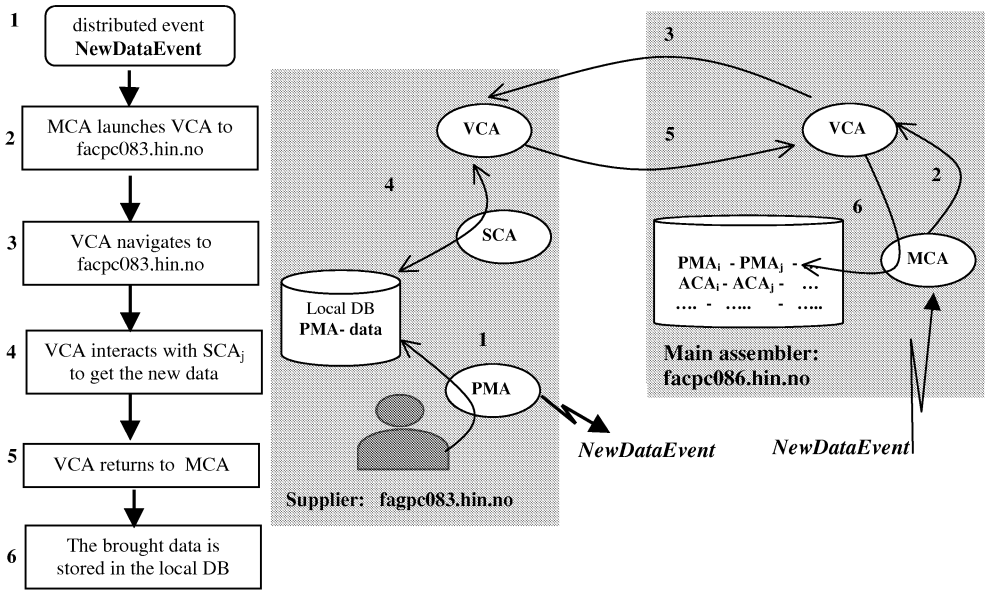
The backbone of the data collection system for the formation phase is based on mobile agents launched by the main assembler frequently visiting supplier enterprises. Rather than being a passive contributor, a supply enterprise may inform the main assembler about the availability of its newest data. Whenever a supplier updates its product data on its web site, provisions can be made to inform the assembler automatically- either by sending an email or by launching a mobile agent from its server. By this mechanism, the main assembler will always be aware of the suppliers' newest data. Another way of providing suppliers' new or updated data is to feed these data into the main assembler's web site provided that the assembler on its web site hosts some "schema" or a form for fill-in. These features are to be added to the prototype in near future.

4.6 THE TESTING PROTOTYPE FOR THE OPERATION PHASE

Based upon the multi-agent model described by the scenario in section 4.2.2, and guided by the architecture described in the section 4.3, we implemented a testing prototype in our laboratory, for the operation phase. Current version of our prototype is implemented with the Concordia (version 1.1) agent development software.

In the testing prototype, the whole system's operation is based on following five asynchronous distributed events:

1. New data (*NewDataEvent*)
2. Change of data (*ChangeDataEvent*)



1 distributed event NewDataEvent
2 MCA launches VCA to facpc083.hin.no
3 VCA navigates to facpc083.hin.no
4 VCA interacts with SCAj to get the new data
5 VCA returns to MCA
6 The brought data is stored in the local DB
VCA
SCA
Local DB PMA- data
PMA
Supplier: fagpc083.hin.no
NewDataEvent
VCA
MCA
PMAi - PMAj - ACAi - ACAj - ...
Main assembler: facpc086.hin.no
NewDataEvent

Figure-4.13: Physical processes of collecting new data from collaborators

3. Privileged update (*PrivilegedEvent*)
4. Maintenance (*MaintenanceEvent*)
5. Refresh data (*RefreshEvent*)

We explain below the implementation details:

4.6.1 The physical processes of collecting new data

When the supplier (fagpc083.hin.no) enters some new data that is relevant to the main assembler, in the supplier's local database, it triggers the asynchronous distributed event *NewDataEvent* after completion of the data entry. By this triggering, the main assembler is automatically notified about the availability of the new data, so the main assembler coordinator agent begins the processes of collecting the new data, see Figure-4.13. Figure-4.13 shows the operations from distributed event triggering to finally placing the new data in the main assembler local database.

The distributed event triggering mechanism

First of all, to receive the distributed event *NewDataEvent*, the main assembler coordinator agent (MCA) must inform the Event Manager about this interest. This is done by sending a *registerEvents* message to the object *EventManagerConnection* and passing an array of Class objects representing the types of events MCA wish to receive and an *EventPost* implementation. The array of Class objects for events are derived from *EventType* and we use *EventQueueImpl* (for the type of implementation of *EventPost*) for asynchronous notification. Given below is the code that is relevant to registration for receiving NewDataEvent.

The program code shown below shows the constructor of MCA; In the constructor first an *EventQueueImpl* object for asynchronous event notification is created and then a direct connection to the Event Manager is made by sending a *makeConnection* message. Finally MCA is registered to receive the event *NewDataEvent*.

```
/*  MCA.java  :::  Main assembler coordinator agent */
…   …
import COM.meitca.concordia.Agent;
import COM.meitca.concordia.event.*;
…

public class MCA implements EventHandler {

private EventPost eventQueue; // This MCA's event queue

// making connection to the Event Manager
private EventManagerConnection  eventManagerConnection;


/* Constructor to MCA */
public MCA() throws EventException {
        …
    try {
         eventQueue = (EventPost)new EventQueueImpl(this);
         Class eventClass[] = {NewDataEvent.class};
         eventManagerConnection = new EventManagerConnection();
         eventManagerConnection.makeConnection(false);
         eventManagerConnection.registerEvents(eventClass, eventQueue);
    } catch (Exception e) {
        throw new EventException(e.toString() + e.getMessage());
      } //catch
    } //try

} //constructor
```

The launching of VCA from MCA

The program code shown below explains how the event *NewDataEvent* is handled once the agent MCA is notified: Once the event is received, MCA launches the mobile agent VCA to the supplier (fagpc083.hin.no).

```
// part of the global declaration in MCA
…
…
Itinerary itinerarySupplier1 = new Itinerary();
itinerary Supplier1.addDestination(new Destination("facpc083.hin.no",
"queryDatabase"));
VCA.setItinerary(itinerary Supplier1);
… …
…

/* The event handler. */

public void handleEvent(EventType event) throws EventException {
     if (event instanceof NewDataEvent) {
        …
        …
        VCA.launch(); //launching VCA to go to the supplier
        System.out.println("VCA is now launched to fagpc083.hin.no");
     }//if

}// handleEvent
```

Triggering the event at the supplier

Now that we know how the event *NewDataEven* is handled by the MCA, let us explore how *NewDataEvent* is triggered in the first place. According to figure-4.13, the event *NewDataEvent* is triggered after some new data is entered into the supplier's local database. Hence, after the data entry into the local database is complete, the agent PMA generates the event to notify the main assembler agent MCA. To do this, the postEvent method of the agent is used.

```
// part of PMA
```

```
…
…
updateDatabase();

//upon completion of the database entry,
makeEventManagerConnection(false);
EventType NewDataEvent = new NewDataEvent ();
postEvent(NewDataEvent);
```

Retrieving new data from supplier database and storing in assembler's database
The programming techniques for interfacing with local database (retrieving from supplier database and storing in main assembler database) is discussed here.

In figure-4.13, Java database connectivity (JDBC) driver is encapsulated as the wrapper agent SCA. With SCA, it is easy for the visiting mobile agent VCA to get the data from the supplier's database. The code is shown below:

First of all, the global import statements of the mobile agent VCA for interacting with a JDBC driver are shown:

```
/*    VCA.java: implementing the mobile agent VCA  */
… …
…
import jdbc.idbDriver;
import jdbc.idbResultsSet;
…
```

Now parts of the VCA program code that is responsible to reading the supplier's local database is shown here:

```
/*  VCA travels to the given destination identified in the itinerary,
and accesses pure Java database (via JDBC). The information is then
taken to main assembler where the results are deposited in the main
assembler's Java database  */

// here are some statements of VCA that is relevant to JDBC access
public class VCA extends Agent {
      Vector newData; // results of SQL query made by VCA

      public VCA() {
      //  Parts of the Constructor of VCA - the mobile agent
      …
      …
      newData = new Vector();  //to keep the results of SQL query
      } //VCA()
```

```
/* Executes a simple SELECT SQL query on the Java database on
   the destination; results are stuffed into newData Vector.
   The queryResultVCA class is used when storing query results
   in Vector  */

   public void queryDatabase() {
      String   SQLstr0 = "SELECT * FROM tester WHERE
                               itemData = \"RECENT\";

      try {
            String url=
             "jdbc:idb:c://jdb191//supplierDB//supplierDB1.prp";
            //Create a URL specifying an JDBC data source name.
            //URL indicates JDBC data source named supplierDB1.
            idbDriver idb1=new idbDriver();
            Class.forName ("jdbc.idbDriver").newInstance();
            Connection con = DriverManager.getConnection(url);

            DatabaseMetaData dma = con.getMetaData ();
            //print driver info on the console
            System.out.println("Connected to " + dma.getURL());

            //Create Statement object to submit SQL statements
            //to driver
            Statement stmt = con.createStatement ();

            try {
            if (stmt.execute(SQLstr0)) {  //Result set available
            ResultSet rs = stmt.getResultSet();
            //Step through the result rows
            System.out.println("Read data :---- ");
            while (rs.next()) {
                  //build QueryResult to hold data for transport
                  queryResultVCA result = new queryResultVCA();
                  //pull the results out of the Statement
                  result.productNo  = rs.getInt(1);
                  result.productName = rs.getString(2);
                  result.delivery = rs.getInt(3);
                  result.amount  = rs.getInt(4);
                  result.pricePerUnit = rs.getInt(5);
                  System.out.println("product = "+
                  result.productNo +
                  " name = "+ result. productName +
                  " delivery = " + result.delivery +
                  " amount= " + result.amount+  " price pUnit = "
                  + result.pricePerUnit);
                  // stuff the result into the newData vector
                  newData.addElement(result);
            } // while
            rs.close();
         } // if
      } catch (SQLException e) {
            e.printStackTrace ();
            } // try-catch

      stmt.close();
```

```
            con.close();
            } catch (java.lang.Exception ex) {
                  System.err.println("Error: " + ex);
                  ex.printStackTrace();
            } // try-catch
      } // public void queryDataBase

…

} // class VCA
```

Finally, the object that is responsible to transporting the data from supplier to assembler is shown below:

```
/* simplified object for data transport  */
class queryResultVCA implements Serializable {
      int         productNo;
      String      productName;
      int         delivery;
      int         amount;
      int         pricePerUnit;
} // class queryResultVCA
```

A CORBA wrapper agent

Wrapper Agent is an important part of the data collection system. For example, in figure-4.10, wrapper agent SCA wraps the database using JDBC driver, expressing the interfaces to access the database. Though it is much easier to encapsulate Java database connectivity (JDBC) driver as the SCA, in some cases we may prefer to wrap a database into a CORBA object instead. This is because of two reasons: 1) SCA is not just a wrapper, it is a coordinator for the supplier- thus has other jobs to do; 2) Making database as a CORBA object also satisfies our intention to represent the database as any (legacy) CORBA object, not just a database.

If we need to implement a CORBA wrapper that will interact with a visiting mobile agent, the integration can be achieved by using the following strategy: First the mobile agent VCA travels to the destination. On the destination, the resource (database) is encapsulated with CORBA-IDL and thus made as a CORBA object.

On the destination, Concordia Service Bridge is used to encapsulate the ORB specific calls to the CORBA object (see Figure-4.14). When the mobile agent (VCA) wants to access the database (CORBA object), the service bridge would locate the CORBA object, retrieve a reference to it, and then delegate method calls down to the CORBA object. By this approach, the agent needs to know nothing about the specifics of the CORBA API. From agent's point of view, it is just accessing services using standard Concordia mechanisms. Also, when both agent and the CORBA object are located on the same computer (as in our prototype), and if the ORB has a "same server" optimization then it may bypass the TCP/IP stack and use shared memory, named pipes, or other inter process communication mechanisms. This will yield much better performance.

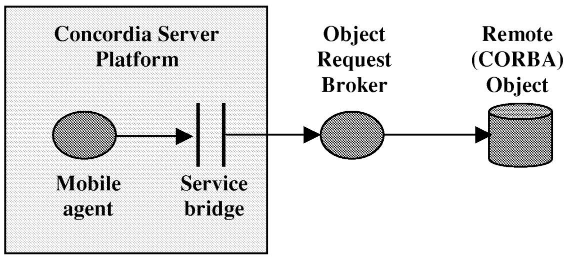


Figure-4.14: Integrating mobile agent with CORBA using the Concordia Service Bridge

4.6.2 The other events

In the previous subsection, we studied how the distributed event *NewDataEvent* is realized. In this subsection, we discuss how the other events are realized. Since realization of theses events are very similar to the event *NewDataEvent*, we do not give any program code.

Change of existing data (*ChangeDataEvent*) about an enterprise: after updating its local database with the new data, the respective functional agent of the supplier executes the distributed event *ChangeDataEvent*, which triggers the MCA to start the update cycle similar to that shown in figure-4.13.

Privileged update (*PrivilegedEvent*) is also similar to change of existing data. The only difference is that, in addition to the main assembler, the new data is sent to specific enterprises specified by the sending enterprise. Of course, main assembler always receive any privileged data, as main assembler is the master of the collaboration.

Since we are only using a supplier and a main assembler (see figure-4.8), a supplier can not send any privileged data to any other enterprise other than assembler; thus this event is not completely programmed and checked.

During the maintenance cycle (triggered by *MaintenanceEvent*), validity of the data in assembler's database is checked. Copy of data about an enterprise is sent to the respective enterprise for verification. This maintenance cycle is triggered by an external real-time driven program (say every four days) or by any applications on the assembler, after finding invalid data in the database.

Refresh cycle (triggered by *RefreshEvent*) is similar to maintenance; the difference is that this cycle is triggered by an enterprise wanting to make sure that its data in assembler's database is valid. In this case, the assembler sends data to this enterprise only. Refresh is normally triggered after the *NewDataEvent* event.

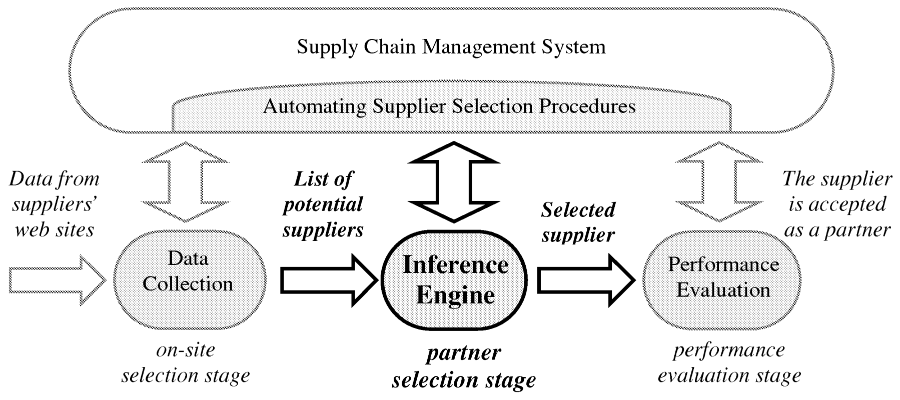
If an enterprise wants to distant itself from collaboration for a shorter period, it triggers *NewDataEvent* with empty data. During the next maintenance cycle, the assembler ignores the enterprise from further collaboration after seeing empty data in its database.

4.7 SUMMARY

The data collection system described in this chapter is the first of the three modules for automating supplier selection procedures. The data collection system sends mobile agents to collects data (or quotes) from suppliers' web sites; to enable this, data provided by suppliers on their web sites are structured-information using XML conforming to a publicly available uniform grammar. After reading the suppliers quotes, the mobile agents then accepts the supplier as a potential supplier and bring backs the data to the main assembler for further scrutiny, only if the supplier data satisfies the conditions sets by the assembler with broad margins. Because of mobile agents selecting suppliers on their web sites, this stage is called the 'on-site' selection stage.

Collecting data from potential suppliers is for the formation phase of an agile virtual enterprise. The data collection system can be extended to play the information infrastructure role during the operation phase too; this is also explained in this chapter.

5. THE INFERENCE ENGINE FOR DECISION MAKING



Supply Chain Management System
Automating Supplier Selection Procedures
Data from suppliers' web sites
List of potential suppliers
Selected supplier
The supplier is accepted as a partner
Data Collection
Inference Engine
Performance Evaluation
on-site selection stage
partner selection stage
performance evaluation stage

Figure-5.1: The inference engine as a part of automating supplier selection procedures

Our design philosophy for automating supplier selection procedures is shown again in figure-5.1, where the module that is relevant for this chapter is highlighted.

From figure-5.1 and from the previous chapter on data collection system, we know that mobile agents bring bids from potential suppliers. Now we need an engine- let us call it the inference engine, that will process the arrived data. Using some set points set by the management, the inference engine should select some suppliers. As for an automated system, the inference engine should be capable of processing large volume of data at high-speed ('real-time').

5.1 TECHNOLOGIES FOR REALIZING INFERENCE ENGINE

The technology for realizing the inference engine should posses the following qualities:

1. Fast processing for real-time application
2. Compact size, should not demand large memory, disk size or extra processing power
3. Easy to build and maintain
4. Easy to integrate with the rest of the supplier selection procedures

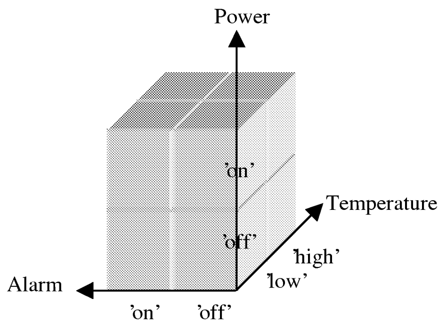
A survey was done to choose the optimal technology to realize the inference engine, satisfying the qualities mentioned above. Among many competing technologies the popular ones like fuzzy logic, and array-based logic were also considered. A brief account of the survey is given below; the results of the survey is also summarized in table-5.1.

5.1.1 Propositional Logic

Propositional logic is concerned with the validation of an argument consisting of a set of propositions split up into a number of premises and conclusions. The boolean logical variables describe the facts in the premises, and the premises themselves describe the system when combined together.

Lets say that a logic system consists of three primitive logic variables, Temperature (with domain values 'low', 'high'), Alarm ('off', 'on'), and Power ('off', 'on').

5.2a:
The space spanned by the primitive logic variables Power, Alarm, and Temperature



5.2b:
The subspace spanned by the combination
((((Temp is 'low') AND (Alarm is 'off')) OR (Power is 'on'))
=> (Power is 'on'))
AND
(((Alarm is 'on') OR (Power is 'off')) => (Power is 'off'))

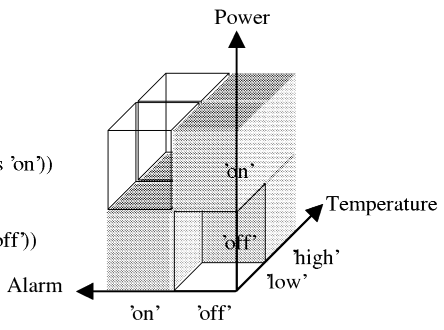


Figure-5.2: Configuration space spanned by the logic variables

Mathematical reasoning approach

By the use of propositional logic, modeling a logic system can be done exactly like modeling a physical system [Bjørke, 2000]. First, the fundamental logic variables (also called primitive logic elements) are identified and each logic variable is assigned an axis; thus the logic variables span the whole universe of discourse (total space), see figure-5.2a. Then the logic variables are connected into premises, thus creating a subspace of the total space, see figure-5.2b. Finally, the premises are combined to form the logic system, connecting subspaces spanned by the premises. There are some differences between the space span by the physical systems and logical systems; logical spaces are always linear and discrete.

By connection, space which do not satisfy the constraints are removed, leaving a smaller space which represents the feasible solution (figure-5.2); this is after Lagrange, who in analytical mechanics developed the free variational method. Thus Lagrange developed the mathematical foundation for the basic procedures for logic modeling discussed in this paper, and it was Pierce who applied these procedures (constraint satisfaction) to logical problems [Møller, 1995].

Advantages and disadvantages of propositional logic
This logic representation is useful in providing formal proofs as it offers clarity. Logic systems modeled with propositional logic is well defined and easily understood [Kusiak, 1997]. Also, by the mathematical approach for modeling logic systems, a Cartesian axis is assigned to each logic variable in the system, generating subspaces spanning all possible states of all the variables, thus providing a *complete representation*. However, there is serious shortcoming of propositional logic that disqualify itself as the technology for realizing inference engine: though propositional logic offers complete systems, the representation is huge; this means, for M boolean logic variables, the resulting space of M axes will contain M^2 subspaces. This exponential growth (also known as 'combinatorial explosion') of the subspaces with increasing number of variables makes the modeling and simulation slower. Thus, propositional logic is not suitable for realizing the inference engine.

5.1.2 Predicate Logic

This is much like propositional logic, but deals with multi-valued logical variables. In the mathematical approach for modeling predicate logic systems, a Cartesian axis is assigned to each logic variable in the system, generating subspaces spanning all possible states of all the variables, thus providing a complete representation; this representation is also huge as for M multi-valued logic variables with a domain of N values, the resulting space will contain M^N subspaces. Advantages and disadvantages of predicate logic are similar for that of propositional logic. Due to the combinatorial explosion, like propositional logic, this technology is not useful for realizing inference engine.

5.1.3 Production Rules

Production rules are in effect subsets of predicate calculus with an added prescriptive component indicating how the information in the rules is to be used in reasoning. A production rule has the following form [Kusiak, 1997]:

IF (condition)
THEN (conclusion)

Mathematical reasoning approach
The basic reasoning approach employed for production rule is searching: starting with a set of facts and look for those rules in which the IF clause matches the facts; this such

rules are found ('hit'), then proceed to the THEN clause. This reasoning is known as 'forward reasoning'. In 'backward-reasoning', searching starts with a set of desired goals and to look for those rules in which the THEN clause (conclusion) matches the goals.

Figure-5.3 shows an example with 6 rules; the example is the same as the one discussed in figure-5.2. As usual for production rule, AND/OR tree is used to illustrate the inference process. In figure-5.3, forward reasoning (or bottom-up search) is used.

The logic system described in figure-5.2 is represented here by the following six rules:

Rule-1:	IF ((Temp is 'low') AND (Alarm is 'off')	THEN (R1)
Rule-2:	IF (R1) OR (Power is 'on')	THEN (R2)
Rule-3:	IF (R2)	THEN (Power is 'on')
Rule-4:	IF ((Alarm is 'on') OR (Power is 'off'))	THEN (R3)
Rule-5:	IF (R3)	THEN (Power is 'off')
Rule-6:	IF ((Power is 'off') OR (Power is 'on')	THEN (Goal)

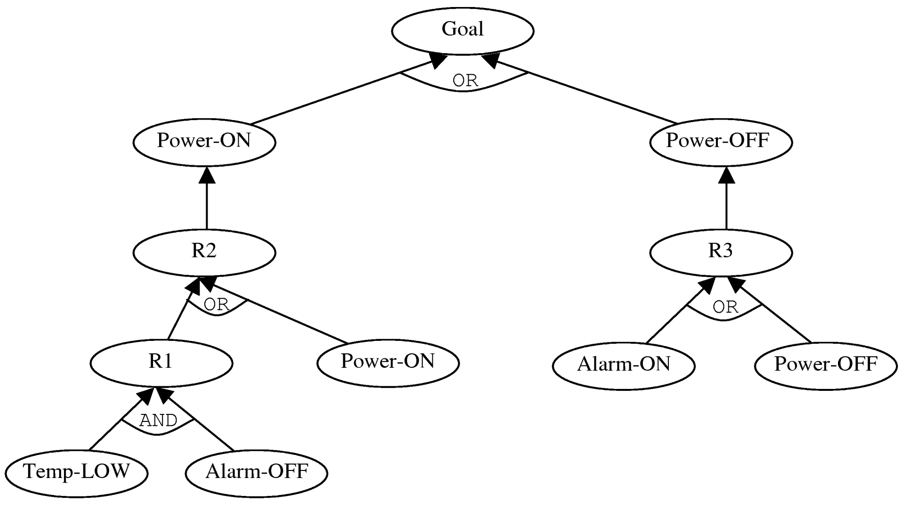


Figure-5.3: Forward reasoning inference process

Advantages and disadvantages of production rules

The simple rules are easy to understand and to modify and extend. However, there are some shortcomings: In production rules, a logic system is evaluated with a couple of 'if-then' statements, taking a linguistic view than a mathematical approach. In this means, for M multi-valued logic variables with N values, there is a need for M^N 'if-then' statements to span all combinations of the variables. Missing any of these statements may cause unexpected results; for a complex logic system of many variables, it is impossible to write so many if-then statements to take care of all possible combinations of variables, thus making the model prone to errors. Besides this possibilities for incomplete system, the reasoning approach based on searching is slow. Thus production rules is not suitable for realizing our inference engine.

5.1.4 Fuzzy Logic

Since Fuzzy Logic is a promising technology to realize the inference engine for automating the supplier selection procedures (see table-5.1 for summary), we do a detailed study about this technology. For further reading about fuzzy logic, refer [Yager and Zadeh, 1991; Tsoukalas and Uhrig, 1997; Adcock, 1993; Meridian, 1997].

Fuzzy logic is a methodology for expressing operational laws of a system in linguistic terms instead of mathematical equations. Systems that are too complex to model accurately using mathematics, can be easily modeled using fuzzy logic's linguistic terms. These linguistic terms are most often expressed in the form of logical implications, such as fuzzy if–then rules. For example, a fuzzy if-then rule (or simply a *fuzzy rule*) looks like:

If delivery_time is LATE, then supplier_preference is LOW.

The terms LATE and LOW are actually sets that define ranges of values known as *membership functions*. By choosing a range of values instead of a single discrete value to define the input parameter "delivery_time", we can compute the output value "supplier_preference" more precisely. Inference mechanism in fuzzy logic is based on fuzzy rules that connect input and output parameters (fuzzy rule base), and the membership functions for input and output parameters.

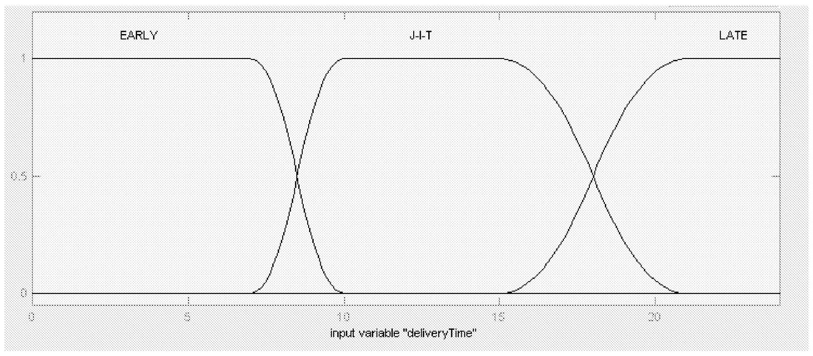
To create an inference engine for supplier selection, first we should develop the membership functions for the input parameters such as "cost", "quality", "delivery_time", and for output parameter "supplier_preference". These membership functions are defined by both a range of values and a degree of membership. The membership functions for an input parameter- "delivery_time" and for the output parameter "supplier_preference" are shown in figure-5.4.

Inference mechanism in Fuzzy Logic

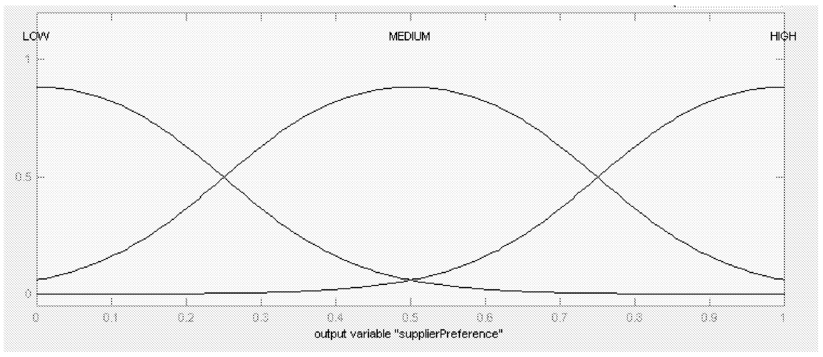
Inference mechanism in Fuzzy logic is implemented in three phases (see Figure-5.5):
Phase-1: Fuzzification phase (converting crisp input value into fuzzy value).
Phase-2: Inference phase (computing fuzzy output value by the fuzzy rules base).
Phase-3: Defuzzification phase (converting fuzzy output value into crisp value).

Fuzzification phase

Fuzzification means mapping an input value into fuzzy membership functions, thus converting the crisp input value into fuzzy value. The degree of membership is found by finding the intersection point of a distinct input value on the horizontal axis with the line defining one or more fuzzy membership functions. For example, when "delivery_time" is 9 (days), "delivery_time" is a member of the function EARLY with a relative membership of 0.24, and also a member of the function J-I-T with a relative membership of 0.76, see figure-5.6.



Membership functions for the input parameter "delivery_time"



Membership functions for the output parameter "supplier_preference"

Figure-5.4: Membership functions

Inference phase

In the inference phase, fuzzy values of the output parameters are computed from the fuzzy input values by the fuzzy rule base. For example, the supplier selection system has the following fuzzy rules, regarding the input parameter “delivery_time”:

If delivery_time is LATE, then set supplier_preference to LOW.
If delivery_time is EARLY, then set supplier_preference to MEDIUM.
If delivery_time is J-I-T, then set supplier_preference to HIGH.

With the known fuzzy values of the input parameter “delivery_time”, the fuzzy values for the output parameter “supplier_preference” can be computed.

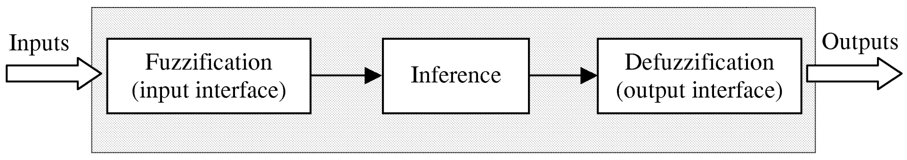

Inputs
Fuzzification (input interface)
Inference
Defuzzification (output interface)
Outputs

Figure-5.5: The three phases of inference mechanism in fuzzy logic

Defuzzification phase

After the fuzzy values for the output parameters are computed by combining the input fuzzy values with fuzzy rule base, the output fuzzy values must be converted back to crisp values. this is done in this defuzzification phase. There are several methods available for defuzzification, such as maximum method and centroid method. In centroid method, the crisp value of the output parameter is computed by finding the value of the center of gravity of the membership function for the fuzzy value.

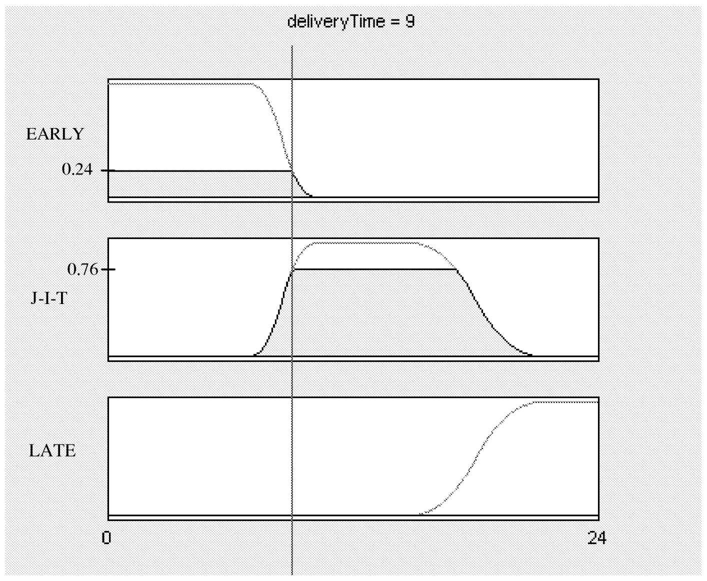


Figure-5.6: Relative membership values

A simple fuzzy inference engine

As an example, a simple fuzzy inference engine was made to compute supplier selection scale based on the delivery time. The fuzzy engine is based on the membership functions for the input parameter "delivery_time" and the output parameter "supplier_preference" (shown in figure-5.4), and the fuzzy rule base given above connecting "delivery_time" "supplier_preference". The supplier selection scale for different delivery time is computed by the inference engine is shown in figure-5.7; according to this figure, when the delivery is EARLY (0-7 days), the supplier selection (supplier preference) scale is about 0.5; when the delivery time increase to J-I-T (10-15) days, the selection scale increases steadily to 0.8, and remains constant for the J-I-T period. When the delivery time further increases into the LATE period, the selection scale steadily decreases, and falls to about 0.17.

Advantages and disadvantages of fuzzy logic

Fuzzy logic is used to solve many problems within manufacturing related areas such as, job shop control, scheduling, robotic arm movement, manufacturing process control, etc. Fuzzy logic is a promising technology to realize the inference engine for the supplier selection automation. Fuzzy logic offers fast inference, offers compact executable code that can be downloaded into micro-controllers for embedded applications. Fuzzy logic is also easy to learn and use.

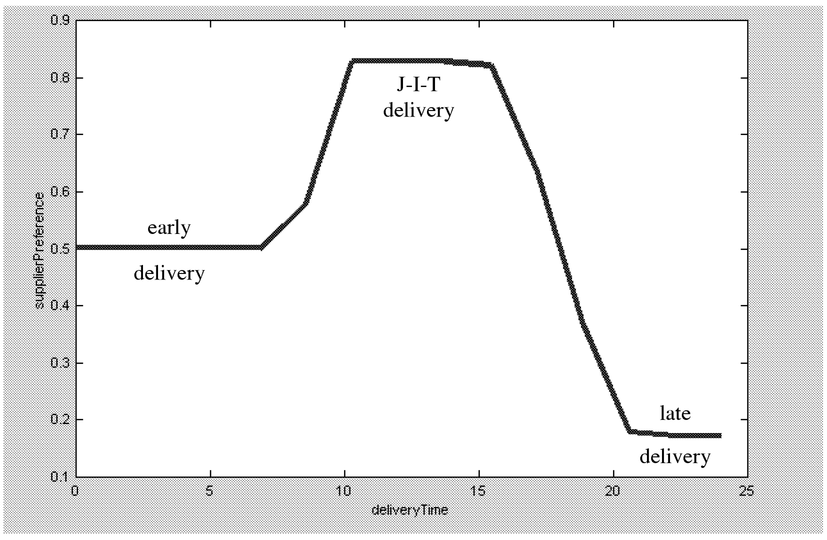


Figure-5.7: Supplier selection scale for different delivery time

One disadvantage of fuzzy logic is tuning; if we want to change the pattern the output parameters are computed from the input parameters, then in-addition to changes in the fuzzy rule base, we may have to change the membership functions of the input and output parameters too. Another disadvantage is that fuzzy logic does not guarantee completeness; it is up to the designer to include all the fuzzy rules connecting all possible combinations between the input and output parameters.

The third disadvantage is the difficulty in generating fuzzy rule base. The fuzzy rules generated for an application must be consistent; they must properly adhere to the process dynamics with no contradictions between the rules. Generating the antecedent (the IF part) of a fuzzy rule is easy; but generating the consequent of a fuzzy rule (the THEN part) is not easy, as it demands deep understanding of the process dynamics.

5.1.5 Array-Based Logic

We know that by predicate logic, the complete representation of M multi-valued logic variables with a domain of N values contains M^N subspaces. This exponential growth of the subspaces with increasing number of variables makes the modeling and simulation slower. Array-Based Logic developed by O. I. Franksen and G. L. Møller avoids this exponential problem by compressing M^N subspaces into $M \ x \ N$ linear representation [Møller, 1995; Franksen, 1979]. Array-based logic also provides mechanisms for operations to operate on the compressed representation in linear time.

In addition to boolean variables and multi-valued variables, array-based logic allows also quantitative (intervals, for example) to be treated as logic variables. There are three types of variables in array-based logic: the nominal logic variables (boolean and multi-valued), ordinal logic variables (e.g. Coordinate is [2,2], [4,2], or [3,3]) and intervals (e.g. Cost is between <50 and 100>).

Technology	*Property (Relation)*	*Inference mechanism*	*Inference cycle time*	*Complete system*	*Implement-ation*
Propositional logic	boolean truth values	modus ponus etc.	slow	yes	not compact
Predicate logic	any predicate	same as propositional	slow	yes	not compact
Production Rules	IF - THEN	searching	slow	no	not compact
Fuzzy logic	fuzzy rules	membership functions & fuzzy rule base	fast	no	compact
Array-based logic	any predicate	geometry	fast	yes	compact, easy to build & maintain

Table-5.1: Summary of the survey on approaches for modeling logic systems

On closer inspection (see table-5.1), array-based logic satisfies all the requirements on the qualities for realizing the inference engine. Therefore we chose array-based logic to realize the inference engine. For detail insight into array-based logic, the Ph.D. dissertation by G. L. Møller should be referred [Møller, 1995]. Table-5.1 shows the results of the survey. In the next subsection, we shall go through the details of realizing array-based logic in MATLAB environment.

5.2 REALIZATION OF ARRAY-BASED LOGIC

We realized ABL with MATLAB simulation system with the following goals in mind:

1. To make computations with natural-language like statements
2. To make a realizations that can be portable to other platforms
3. For easy implementation of the inference engine.

The main functions in the MATLAB realization of the array-based logic technology are explained below with the help of six definitions. The example codes given below are very relevant for implementing the inference engine.

5.2.1 Propositional logic functions

All the logic variables that are used in a system are to be declared first. In our realization, the function `element` is used for declaration. Though there is no logic element exist in reality, we consider logic variables as virtual elements just to be in-accordance with the terminology for modeling physical systems in manufacturing systems theory. Relevant to the function `element` is the function `assign`; this function changes the values of a logic variable.

E.g. to define a multi-valued logic variable 'Delivery_Time' with a domain of three values 'early', 'just-in-time', and 'late', we use,
Delivery_Time = element('n', {'early', 'just-in-time', 'late'}, {'early'}, 'Delivery Time');
The first argument 'n' indicates that the variable is of nominal type. The second group of input argument are values (of domain), the third group is the default values selected at the time of declaration (in this example, default value is 'early'), and the final input argument is the label or name of the variable. After declaring a logic variable, we could change the values of the variable with the function `assign`;
LateDelivery = assign(Delivery_Time, {'late'});

■ ***Definition 5.1: Basic operations***
Any logic systems can be developed by applying the following four basic operations on variables: disjunction (V), direct-implication (=>), nand, and converse-implication (<=). These four operations are known as the Klein four group. Other logic operations can be derived from these four basic operations. In our realization, the functions for these four operations are `disjunct`, `dimp`, `nand`, and `cimp` respectively. ■

E.g. If Premise1 = (LateDelivery => Reject), then we write,
Premise1 = dimp(LateDelivery, Reject);

■ ***Definition 5.2: Colligation: Removing duplicate variables***
If the same variable occurs more than once in a premise or in a combination of premises, then duplicate axes will be found in the configuration space. The process of removing superfluous axes without losing any information is called Colligation. The function that performs colligation is `fuse`. ■

E.g. if System = (Premise1 V Premise2), where
Premise1 = dimp(LateDelivery, Reject), and
Premise2 = dimp(HighCost, Reject),
Then the System will contain two copies of the logic variable Reject (or mathematically, two axes for Reject). A duplicate Reject is removed (or the axes are fused together) by,
System = fuse(System);

Table-5.2 shows the main propositional logic functions and some utility functions.

Propositional logic functions	
Basic operations	
disjunct	OR operator
conjunct	AND operator
dimp	direct implication
bimp	binary implication (equivalence)
cimp	converse implication
exor	exclusive OR
nand	NOT-AND operator
invert	NOT operator
Format change	
affirmative	to extract valid configuration space from whole space
unpack	opposite of affirmative
Deductions	
ared	abductive (AND) reduction
dred	deductive (OR) reduction
fuse	to remove superfluous axes (variables)
state	to find state of a system (or outputs) for a given inputs
Utility functions	
element	to create a logic variable (or primitive logic element)
assign	to assign new values to a multi-valued logic variable
domain	to assign new domain to an interval variable
print	to printout a relation (variables, premises, combined system)

Table-5.2: The main propositional logic functions and some utility functions

5.2.2 Array-based logic functions

The objective of array-based logic is to make a technology satisfying the three requirements completeness, compactness, and faster simulation. By using propositional logic functions, complete solutions are obtained. However the resulting systems are not compact therefore computations are slow. Array-based logic fulfills compactness and fast processing by working on the legal combinations only (shaded subspace areas in figure-5.2b), and by not working on the whole space. The following definitions present the main functions of our realization. of array-based logic.

■ ***Definition 5.3: Compressed representation***
Compressed representation is to keep the relation (premises, subsystems, or system: see figure-5.8) to a minimal size without loosing any information. The function used for compression is `compress`. ■

In compressed form, functions like `join`, `deduct`, etc. make use of the compressed (compact) representation for faster computation. The function join connects premises together via the common variables they posses; the resulting relation (subsystem, or system) will be in compressed form. Compression technique is similar to the Karnaugh map (K-map) reduction done by-hand in digital electronics.

In addition to boolean variables and multi-valued variables, array-based logic allows also quantitative (intervals, for example) to be treated as logic variables. There are three types of variables in array-based logic: the nominal logic variables (boolean and multi-valued), ordinal logic variables (e.g. Coordinate is [2,2], [4,2], or [3,3]) and intervals (e.g. Cost is between <50 and 100>).

■ ***Definition 5.4: Intervals as logic variables***
Array-based logic facilitates intervals to be treated as logic variables. An interval variable may contain many intervals, each of which may be true or false. ■

To create an interval, the function `interval` is used.
E.g.: LowerInterval = interval('ge', 85, 'lt', 98).
This means, the LowerInterval is greater than or equal to 85, and less than 98.

An interval variable is created using the function `element`.
E.g.: InputPrice = element('i', {LowerInterval, UpperInterval}, 'Input Price'),
where the first argument 'i' indicates that the variable to be created is an interval variable, and the final argument is a label of the variable.

■ ***Definition 5.5: Deducing conclusions***
Deduction (or inference) is to draw conclusion from a connected system. Deduction is performed by the function `deduct`, which makes the OR - projection of all the axes complementary to the variables concerned, on the axes of the variables. ■

The final definition is about the state of a system.

■ ***Definition 5.6: state of system***
The state of a system is the information required of the system to uniquely determine an output for an input to the system. The output is a vector of output variables which is computed from the input vector of variables and the system (see figure-5.8), using the function `state`. ■

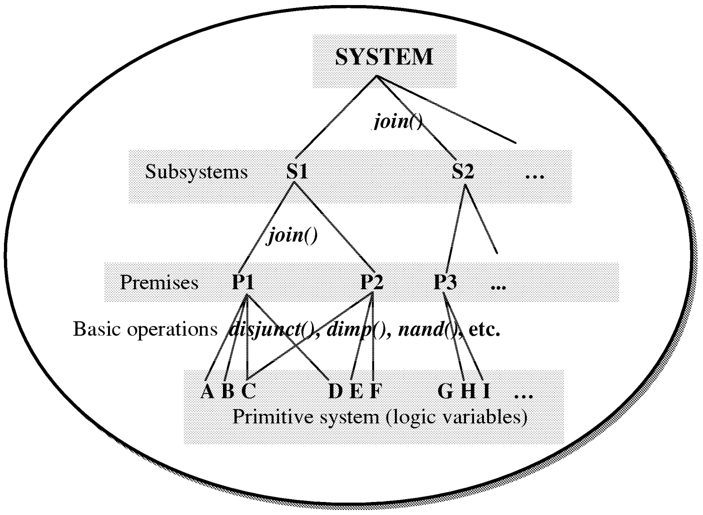

SYSTEM
join()
Subsystems
S1
S2
...
join()
Premises
P1
P2
P3
...
Basic operations disjunct(), dimp(), nand(), etc.
A B C
D E F
G H I ...
Primitive system (logic variables)

Figure-5.8: The system perspective of a modeling a logic system

Allowing quantitative variables to be treated as logic elements facilitates numerous advantages in modeling complex logic systems. Use of propositional and array-basic logic functions will become clear in the next section where implement the inference engine. See [SABL, 2000] for more elaborate explanation of the logic functions. Table-5.3 shows the main functions of our realization.

5.3 REALIZING INFERENCE ENGINE WITH ARRAY-BASED LOGIC

In this section, we shall develop an inference engine, which is to be used to evaluate supplier quotations. Subsections 5.3.1 - 5.3.4 explains the modeling methodology. In the modeling approach (see figure-5.8), first the primitive logic elements (logic variables) are identified. Then these primitive elements are grouped into premises using the logic operators like disjunct, dimp, etc. Finally, the premises are joined to make the compete system. Only the final model (given in subsection 5.3.5), a compact representation of the complete model is needed to evaluate the supplier quotations. Only this compact representation will be converted into C++ language code and then compiled into an executable module (implementation).

Array-based logic functions	
Compression techniques	
compress	to make compact model of a system
expand	opposite of compress
comsize	complement sizes of variables in a system
domsize	domain sizes of variables in a system
Interval	
interval	to create an interval
union	union (combination) of intervals
complement	to complement an interval
intersection	intersection of intervals
Combination	
join	to combine relations (premises) through common variables

Table-5.3: The main array-based logic functions of the toolbox

Let us assume that the mobile agents bring many quotations from potential suppliers. These quotations are fed into the inference engine, which decides whether to select the supplier (with or without reservations) or to reject the supplier (see figure-5.9). We assume that the quotations consist of only four data (figure-5.9). The *price* for a specified number of units let us say one thousand units. The *quality* of the product denoted by the quality index. Product *arrival time* (or delivery time or delay) in days, time taken for the consignment to arrive at the enterprise, once the order is submitted to the supplier. And finally, the *quantity* (number of units) the supplier can supply within the quoted delivery time. These four key performance indicators are chosen to conform the Kanban manufacturing paradigm: right price, right time, right quantity, and right quality. To make decisions on the data, the management keeps four set points for each of the data in supplier quotations. These set points are varied (fine-tuned) by the management to make the selection process agile; for example, if all (or most) suppliers fail in the selection process then the set points are relaxed a little, to make some suppliers pass the selection process.

Figure-5.10 shows the logic variables and the premises that make up the complete system. The first four premises deal with the input values. The input (numeric) values for price, product arrival (delivery time), quantity, and quality are used to assign values to some interim logic variables. In effect, the first four premises are about converting interval variables into nominal variables. Premises 5 to 7 uses the interim logic variables to compute values to the output variable.

5.3.1 Premises 1-to-4: dealing with the input values

Premises 1 to 4 deal with the input values about the product in the supplier quotation. The input values are named as inputPrice, inputArrival, inputQuantity, and inputQuality.

Premises 1 to 4 is to convert the input numerical values into logic variables named price, arrival, quantity and quality respectively.

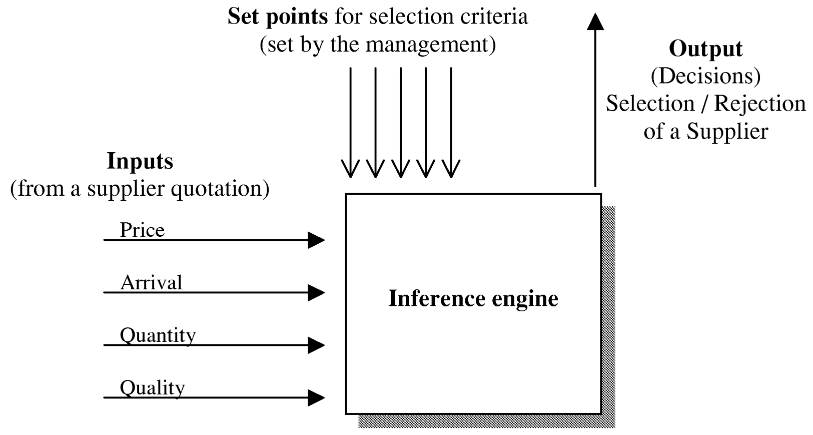


Figure-5.9: Inference engine for supplier selection

Dealing with input price

If the input price is greater than the set point for price, then the price is 'high'. If the input price is less than or equal to the set point for price, then the price is 'fair'. To formulate this logic statement, we need two logic variables. A multi-valued logic variable price with the domain values of 'fair' and 'high', and an interval logic variable with two intervals, one interval between minimum possible price to set point and the other interval between set point to maximum possible price. MATLAB codes for creating the variables are shown below. MATLAB codes start with a MATLAB prompt '>'. In MATLAB, the text that follows the '%' mark are comments, that is, not executable.

First we create the logic variable price.

```
> %declaring variable price with domain value 'fair' &
'high'
> price = element('n',{'fair', 'high'},{}, 'price');
```

Before creating the interval variable InputPrice, we have to assign a value to the set point for price. Let us assume the current value for set point for price is 98 currency units. Let us also assume that minimum possible value for price is 85 and the maximum possible value is 120.

```
> PriceSetPoint = 100; %selection threshold on price
> MinPrice = 85; MaxPrice = 120;
```

Creating two intervals, the lower interval and the upper interval.

```
>    LowerInterval    =    interval('ge',    MinPrice,    'le',
PriceSetPoint);
>   UpperInterval    =   interval('gt',   PriceSetPoint,'le',
MaxPrice);
```

Creating the interval variable InputPrice for input price:

```
> InputPrice = element('i', {LowerInterval,...
    UpperInterval},'Input Price');
```

Finally, the premise-1:

```
> % (PriceIsFair) if and only if (FairPriceRange)
> FairPriceRange = assign(InputPrice, LowerInterval);
> PriceIsFair = assign(price, {'fair'});
> Premise_1 = bimp(FairPriceRange, PriceIsFair);

> % function bimp() is used for "if and only if"
```

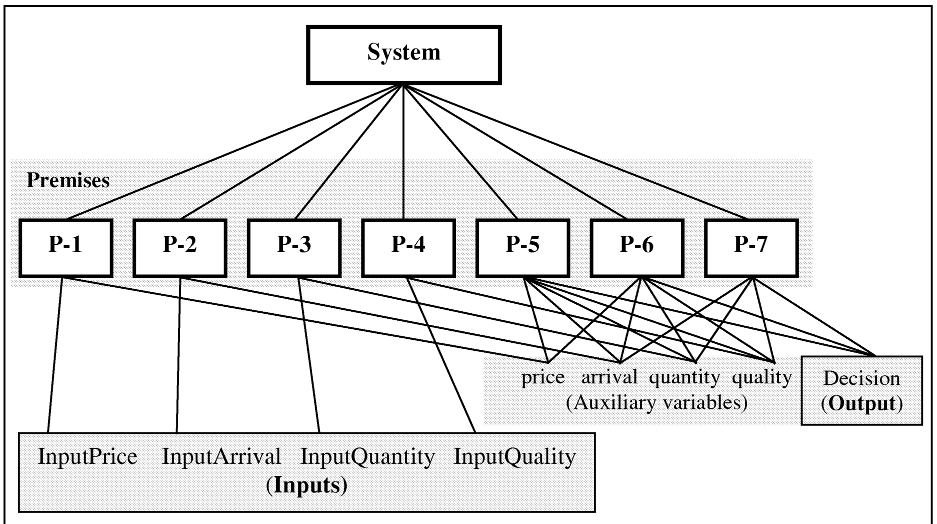

System
Premises
P-1
P-2
P-3
P-4
P-5
P-6
P-7
price arrival quantity quality
(Auxiliary variables)
Decision
(Output)
InputPrice InputArrival InputQuantity InputQuality
(Inputs)

Figure-5.10: Components of the logic model for supplier selection

Dealing with product arrival (or delivery time)

If the delivery time given in the supplier quotation is between the minimum possible delivery time and the first set point for delivery time, then the product arrival is 'early'. If the delivery time is between the first and second set points, then the product arrival is 'just-in-time'. On the other hand, if the delivery time is between the second set point and the maximum allowable delay, then the product arrival is 'late'. To formulate this logic statement, we again need two logic variables. A multi-valued logic variable arrival with the domain values of 'early', 'jit', and 'late'. And an interval logic variable InputArrival with three intervals. The first interval (LowerInterval) is between the minimum possible delivery time to set point-1, the second interval (MiddleInterval) is between the set points, and the third interval (UpperInterval) is between set point-2 and maximum allowable delivery time.

Formulating the premise that deals with the inputArrival is very similar to premise-1 for inputPrice. The only difference is that, delivery has three intervals:

First, creating the variable arrival

```
> arrival =
element('n',{'early','JIT','late'},{},'arrival');
```

Assigning values to the set points for arrival

```
> EarliestArrival = 7; LatestArrival = 21;
> ArrivalSetPoint1 = 10; ArrivalSetPoint2 = 15; %limits
```

Creating the three intervals, the lower, middle and the upper intervals.

```
> LowerInterval = interval('ge', EarliestArrival, 'le',...
      ArrivalSetPoint1);
> MiddleInterval = interval('gt', ArrivalSetPoint1,...
      'le', ArrivalSetPoint2);
> UpperInterval = interval('gt', ArrivalSetPoint2,...
      'le', LatestArrival);
```

Creating the interval variable InputArrival for delivery time:

```
> InputArrival = element('i', {LowerInterval,...
    MiddleInterval, UpperInterval},...
    'Input Arrival');
```

Towards declaring the premise-2:

```
> EarlyArrivalRange  = assign(InputArrival, LowerInterval);
> ProperArrivalRange = assign(InputArrival,
MiddleInterval);
> LateArrivalRange   = assign(InputArrival, UpperInterval);

> ArrivalIsEarly = assign(arrival, {'early'});
> ArrivalIsJIT   = assign(arrival, {'jit'});
> ArrivalIsLate  = assign(arrival, {'late'});

> % (ArrivalIsEarly) if and only if (EarlyArrivalRange)
> % (ArrivalIsJIT) if and only if (ProperArrivalRange)
> % (ArrivalIsLate) if and only if (LateArrivalRange)
> Premise_2A = bimp(EarlyArrivalRange, ArrivalIsEarly);
> Premise_2B = bimp(ProperArrivalRange, ArrivalIsJIT);
> Premise_2C = bimp(LateArrivalRange, ArrivalIsLate);
```

Finally, the premise-2:

```
> Premise_2 = join(Premise_2A, Premise_2B, Premise_2C);
```

Dealing with shippable quantity and quality

Premise-3 for quality and premise-4 for quantity are formulated very similar to that of premise-2. For brevity, the MATLAB codes for these formulations are not shown here.

5.3.2 Premise - 5: Accepting a bid

The logic variables price, arrival, quantity, and quality are used to compute premise-5. Premise-5 is about the conditions for accepting a supplier. A supplier should be selected if and only if all four inputs values are within the acceptable regions, like price is 'fair', delivery is 'just-in-time' or 'early', quality is 'moderate' or 'superior', and quantity is 'ok', or 'more'.

First the output logic variable Decision is created.

```
> Decision = element('n', {'reject', 'select','memo'},...
      {},'Output Decision');
```

Now the acceptable conditions,

```
> AcptPRI = assign(price, {'fair'});
> AcptQUA = assign(quality, {'moderate','superior'});
> AcptARR = assign(arrival, {'jit','early'});
> AcptQNT = assign(quantity,{'ok','more'});
> AcptCondtion = conjunct(AcptPRI, AcptQUA, AcptARR,
AcptQNT);
```

For these acceptable inputs, the decision is 'select'.

```
> Action = assign(Decision, {'select'});
```

Finally, the premise-5 for accepting a bid (selecting a supplier),

```
> % (output decision is 'select') if and only if
> % ((price is 'fair') AND (quality is
           'moderate'/'superior')
> % AND (delivery is 'JIT'/'early') AND (amount is
           'ok'/'more'))
> Premise_5 = bimp(AcptCondition, Action);
```

5.3.3 Premise - 6: Rejecting a bid

A supplier should be rejected if any one of the following conditions is met: either price is 'high', or arrival is 'late' or quality is 'inferior' or quantity is 'less'.

The conditions for rejection,

```
> RejtPRI = assign(price, {'high'});
> RejtQUA = assign(quality, {'inferior'});
> RejtARR = assign(arrival, {'late'});
> RejtQNT = assign(quantity,{'less'});
> RejtCondtion = disjunct(RejtPRI, RejtQUA, RejtARR,
RejtQNT);
```

For these bad inputs, the action (decision) is 'reject'.

```
> Action  = assign(Decision, {'reject'});
```

Finally, the premise-6 for rejecting a supplier,

```
> %(Output Decision is 'reject') if and only if
> % ((Price is 'high') OR (quality is 'inferior')
> %   OR (arrival is 'late') OR (quantity is 'less'))
> Premise_6 = bimp(RejtCondition, Action);
```

5.3.4 Premise - 7: Notifying about the deviation

Though some of the input values are acceptable, a notification should be sent to the supplier about deviations. For example, if the product is to arrive earlier than anticipated, then even though it will not cause any production delay, early arrival will cause some additional inventory costs due to storage. Similar situation will arise in the case of 'more' products (quantity is 'more') sent by the supplier than expected. In this situation, in addition to selecting the supplier, a memo is also sent to the supplier.

A memo is to be sent to the supplier if any one of the following conditions is met: either the product arrival is 'early', or the quality is 'superior' than expected, or the quantity is 'more'.

The conditions for sending a memo to the supplier,

```
> MemoQUA = assign(quality, {'superior'});
> MemoARR = assign(arrival, {'early'});
> MemoQNT = assign(quantity,{'more'});
> MemoCondtion = disjunct(MemoQUA, MemoARR, MemoQNT);
```

For these inputs, the action (decision) is 'memo'.

```
> Action  = assign(Decision, {'memo'});
```

Finally, the premise-7 for sending memo to a supplier,

```
> %(Output Decision is 'memo') if and only if
> % (quality is 'superior') OR (arrival is 'early')
> %  OR  (quantity is 'more'))
> Premise_7 = bimp(MemoCondition, Action);
```

5.3.5 The combined system

The system is the combination of the seven premises. That is,

```
> System = join(Premise_1, Premise_2, Premise_3,...
       Premise_4, Premise_5, Premise_6, Premise_7);
```

When we join the seven premises, the function join removes duplicate variables in the combined system, and leaves the combined system in the compressed form (the function join fuses (removes) duplicate variables while combining, see definition 5.3). However, we used the four auxiliary variables (price, arrival, quantity, and quality) only to convert the input values to logic variables and to compute the output. Therefore, in the final model, let us remove the auxiliary variables.

```
> %deduce (or extract) only the four inputs and the output
> %from the system
> Inputs  = [InputPrice InputArrival InputQuantity
InputQuality];
> Outputs = Decision;
> SYSTEM_F = deduct([Inputs Outputs], System);
```

The final system (SYSTEM_F) is very compact and complete. This is the core of the inference engine. Because it is compact, the decision made by the inference engine is fast.

5.3.6 Simulations on the combined system

Let us input some sample values to the inference engine.

```
> InputPRI = assign(InputPrice, 90);  %US$
> InputARR = assign(InputArrival, 8);  %days
> InputQNT = assign(InputQuantity, 2350);  %units
> InputQUA = assign(InputQuality, 8.2); %quality index

> TestInputVector =  [InputPRI InputARR InputQNT InputQUA];
> output = state(TestInputVector, SYSTEM_F);
> print(output);
```

The output printed on the screen is:

```
** Output Decision ** : select : memo :
```

This means, for the given input values and for the given set points, the supplier is selected and a memo should be sent.

5.4 SUMMARY

The inference engine described in this chapter is for selecting the best supplier from a list of potential suppliers. Selection of the best supplier from the pool of potential suppliers is done in the sub-stage called the partner selection stage. The inference engine is realized with array-based logic.

Array-based logic is chosen to realize the inference engine because, array-based logic offers fast computation, compact code, and complete solution. Fuzzy logic is another technology that can be used for realization of the inference engine, but fuzzy logic does not guarantee complete solutions.

We added the computing with words capability to array-based logic when we wrote array-based logic in MATLAB system; we call the new toolbox of logic functions- the structured array-based logic. With structured array-based logic, not only objective factors but subjective factors too (factors that can not be easily quantified) can also be included for evaluating the best supplier out of the pool of suppliers. However, for brevity, only the critical factors like price, delivery and quality are considered for selection by inference engine.

6. PERFORMANCE EVALUATION

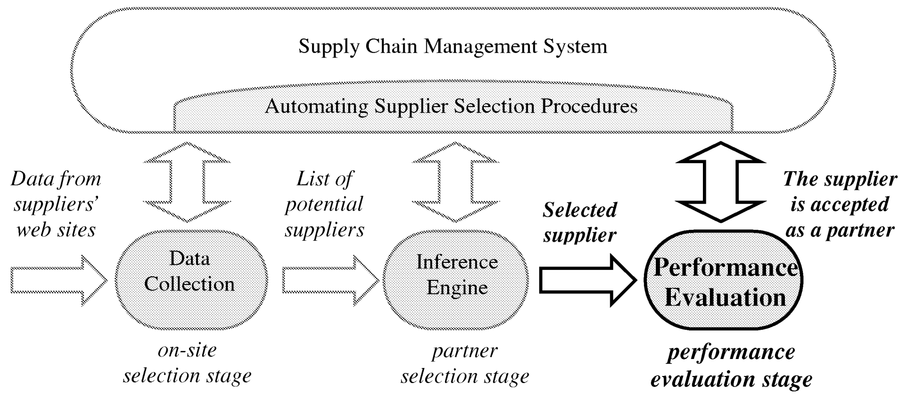

Supply Chain Management System
Automating Supplier Selection Procedures
Data from suppliers' web sites
Data Collection
List of potential suppliers
Inference Engine
Selected supplier
Performance Evaluation
The supplier is accepted as a partner
on-site selection stage
partner selection stage
performance evaluation stage

Figure-6.1: Performance evaluation as a part of automating supplier selection procedures

Performance measurement of supply chain is a very important part of the strategic supply chain management system of an agile virtual enterprise. This is because, the management need a transparent and real-time view of the supply chain so that they can quickly react to the changing market conditions ('agility'), by associating with new collaborating enterprises replacing some existing ones who are performing less satisfactorily ('virtual enterprise'). It is important for management of the agile virtual enterprise to rely on effective tools for performance measurement of the supply chain; in today's competitive environment, management simply cannot afford to wait for a long time for the information they need.

This chapter is about performance evaluation of a supply chain after the introduction (incorporation) of a supplier into collaboration (as a part of supply chain). By taking measurements of supply chain performance and by comparisons, it can be made sure that the selected supplier will indeed perform optimally in collaboration; that is, the selected supplier is the optimal supplier.

Referring to figure-6.1, the module for performance evaluation is shown in connection with the rest of the modules for automating supplier selection procedures. As explained in the introductory chapters, it is the mobile agents of data collection system that bring data (supplier quotations or bids) from potential suppliers. Then the data are processed, or in effect filtered, by the inference engine, which makes decisions on whether to select the supplier as it is, to select it with conditions (or reservations), or to reject it. In the third stage, the selected supplier is then put in the agile virtual enterprise environment as a collaborator and simulation are done to see how the new partner is performing in collaboration. This kind of simulation in collaborative environment is done by the module for performance evaluation (called the performance evaluation engine).

The module can be used in two modes. *Supplier selection mode*: as explained above, during the supplier selection, this module is used to evaluate how the potential supplier performs in collaboration ('performance *evaluation*' of a potential supplier). In the *stand-alone mode*, which has nothing to do with the supplier selection, the module can be used by the supply chain management to measure the performance of the supply chain ('performance *measurement*' of the supply chain). Clearly, 'performance evaluation' (of a potential supplier) is done during the re-configuration phase of the agile virtual enterprise, whereas 'performance measurement' (of supply chain) is done often during the operation phase of the agile virtual enterprise. In summary, performance evaluation engine enables:

- analysis of current supply chain
- design alternative supply chains with new/potential supplier(s); perform "what-if" simulations to predict the performance of new supply chain configurations with new supplier

Though the performance evaluation engine can be used in two modes, we shall emphasize on the 'supplier-selection mode' as this is more relevant to this dissertation.

6.1 SOME ISSUES IN SUPPLY CHAIN PERFORMANCE EVALUATION

We shall go through some issues that are related to performance evaluation of a supply chain. The issues discussed here are the key performance indicators and push/pull material flow.

6.1.1 Key Performance Indicators

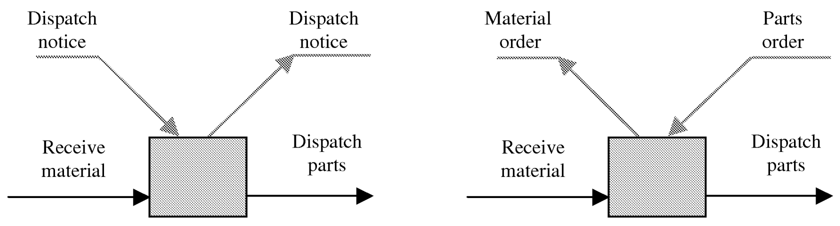
In performance evaluation, in order capture critical quantitative data and qualitative insight into the supply chain, the selection of key performance indicators (KPI) must be limited to a few most important ones. We have selected *cost* (right price), *quantity* (right amount), and *delivery time* (right time) as the key performance indicators.

6.1.2 Push and Pull material flow control systems

When modeling supply chain, frequently, distinction is made between two kinds of material flow control systems, the push systems (e.g. MRP) and pull system (e.g. Kanban operated JIT). However, most practical supply chains consist of both push and pull systems [Bonney et. al., 1999]. Fresh fruits and vegetable production systems is a good example for pull system, whereas production and sales of some custom made specialized instruments are typical example for pull system. The Toyota system, though originally invented the pull system, uses a push system for distributing produced vehicles; but uses Kanban for input raw material and other parts. Thus, Toyota is a good example for hybrid push/pull type [Bonney et. al., 1999].

A fundamental difference observed in push and pull systems is the direction of information flow; in push systems, the information flow is in the same direction as the material flow. Whereas, in pull systems, the information flow is in opposite direction to the material flow. Figure-6.2 shows the differences between push and pull systems, for a part supplier.

The advantages and disadvantages of push and pull systems and the similarities and dissimilarities between them are out-of scope of this work, thus interested reader is referred to [Bonney et. al., 1999; Bonney, 1996]. For simplicity, the examples shown in this paper are based on pull system principle only. With some minute changes, the model versions for push systems can be obtained.



2a: The push system for a part supplier 2b: The pull system for a part supplier

Figure-6.2: The fundamental difference between push and pull systems

6.2 EXISTING APPROACHES FOR PERFORMANCE EVALUATION

In this section, we shall go through some existing approaches for measuring performance of a supply chain. There are a number of approaches available for performance measurement of supply chain; some of them are [Lapide, 2000]:

1. The Balanced Scorecard
2. Supply-Chain Operations Reference (SCOR) Model
3. Activity-Based Costing
4. Economic Value Added

Given below is a short review of these approaches; it must be emphasized that these approaches must be studied within the context of:

- whether these approaches will be useful for selecting a supplier by evaluating the performance of supply chain under the supplier's influence,
- whether any of these approaches could be automated.

6.2.1 The Balanced Scorecard

The Balanced Scorecard concept has been developed in the early 1990s by Robert Kaplan and David Norton as an alternative for traditional performance evaluation techniques [Kaplan and Norton, 1992]. The balanced scorecard combines the strategic planning and goal setting with processes for developing qualitative and quantitative performance measures and benchmarks [Kaplan and Norton, 1992; Lapide, 2000]. In this approach, performance measures are made on the four perspectives: Financial perspective (e.g. cost), customer perspective (e.g. on-time delivery), internal business perspective (e.g. manufacturing planning), and innovation and process improvement perspective (e.g. certified employees).

The main difference between the balanced scorecard approach over the other approaches (e.g. economic value-added) is that balanced scorecard focuses on the indicators (both short-term and long-term indicators) of future performance. Most traditional measures such economic value-added are financial measures, describing yesterday's strategy, as today's financials reflect the performance a year ago. The balanced scorecard achieves a balance between these lag indicators and the lead indicators that need to be focused on to make things happen [Marquardt, 1997].

Though balanced scorecard is used for performance measurement of supply chain, it is not specifically developed for this purpose. The approach is very durable, but demands extensive man-power and time; the first and key step in this approach is to build the scorecard describing the enterprie's strategy, which requires the knowledge of the employees, assets and the infrastructure; enterprises may need six to nine months to work on the issue of describing the enterprise's strategy [Marquardt, 1997]. Thus, the balanced scorecard approach is accepted as a efficient approach also for performance measurement of supply chain, it is not an effective tool for supplier selection. It can not be automated easily either.

6.2.2 Supply-Chain Operations Reference (SCOR) Model

Unlike the balanced scorecard approach which deals with executive enterprise-level measurement of performance with the aim of achieving strategic goals, the Supply Chain Council's (SCC) SCOR model directly addresses on measuring overall supply chain performance.

SCOR is a high-level methodology for describing supply chain components for suppliers, manufacturers, distributors, and customers; SCOR spans the entire supply chain, from the supplier's supplier to the customer's customer. SCOR provides a process reference model of supply chain metrics based on time metrics, cost metrics, service/quality metrics, and asset (inventory) metrics [SCC, 2000].

SCOR is an important development in supply chain management practices because it facilitates strategic planning, tactical planning, and benchmarking by communicating a supply chain in a standard (nonproprietary), industry-neutral format. In short, SCOR provides a set methodology for representing supply chains and analyzing supply chain performance and has become the de facto supply chain standard [SCC, 2000; Gensym, 2000].

SCOR model is a useful supply chain management practices, but it is much more than selecting suppliers. SCOR can also be automated, some automated models for supporting e-commerce and e-business are available, e.g. *e*-SCOR [Gensym, 2000].

6.2.3 Activity-Based Costing

Activity-based costing is an information system that maintains and processes data on an enterprise's activities and products. It identifies the activities performed, traces cost to these activities, and then uses various cost drivers to trace the cost of activities to products.

The activity-based costing approach allows better evaluation of the productivity and costs of a supply chain; For example, to capture the costs of activities in a supply chain, the processes of a supply chain are broken down into measurable activities, and resource requirements (costs and time) are identified; by costing the various activities performed in the processes, it is easy to see how the resource requirements change if one changes the processes (supplier, manufacturing procedure, or distributor).

With an activity-based cost model, it is easy for the management to take decisions because all of the costs for processes are known. It is easy to conduct "what if" analysis, to evaluate the outcome of changing suppliers. The potential of the activity-based costing tool to assist management in daily operational decisions is powerful. However, activity-based costing is not designed to trigger automatic decisions; it is designed to provide more accurate information about production and support activities and product and service costs so that management can focus its attention on what is most effective and efficient [Shank, 1993]. Thus, automating models based on activity-based costing is not feasible. Also, success of implementations based on activity-based costing is not guaranteed, as implementation requires stable operating points on government regulations, complete cost information, and a good experience in cost measurement and cost management. All these information are too much of an effort if the performance evaluation of a supply chain is just for supplier selection.

6.2.4 Economic Value Added

Economic value-added is a measure that is being used for evaluating an enterprise's (say, a supplier's) value-added contributions within a supply chain.

Though useful for assessing long-term shareholder value, economic value-added measures are less useful for measuring detailed supply chain performance [Lapide, 2000]. These metrics can not be easily automated (e.g. for e-commerce), nor simple enough for measurement with the aim of evaluating suppliers.

6.3 NEWER APPROACHES FOR PERFORMANCE EVALUATION

We have seen in the previous section that the existing approaches for performance measurement of supply chain with the aim of evaluating new suppliers are not good enough because, either they can not be automated, or they are too complex.

We devised an effective approach (the performance evaluation engine) for automated performance measurement of a supply chain with the aim of evaluating new suppliers; the approach is based on manufacturing systems theory and Petri nets. The tool that is developed for this approach is called AgileSIM. AgileSIM is developed for MATLAB simulation environment. In the following subsections, we introduce manufacturing systems theory and Petri net.

6.3.1 Manufacturing Systems Theory

The model of supply chain that is developed by our approach has two different levels of detail. The higher-level model shows the flow of material between collaborating enterprises; the basis of the higher-level model is *connection*; the higher-level model (also called the 'connected system') is obtained by the manufacturing systems theory approach. The lower-level model represents the details of individual enterprises which gives rise to the higher-level behavior. The lower-level model is a Petri net. Higher-level model, lower-level model and converting a higher-level model into a lower-level model are explained in-detail later. But right now, let us understand the principles of manufacturing systems theory that are basis of the higher-level model.

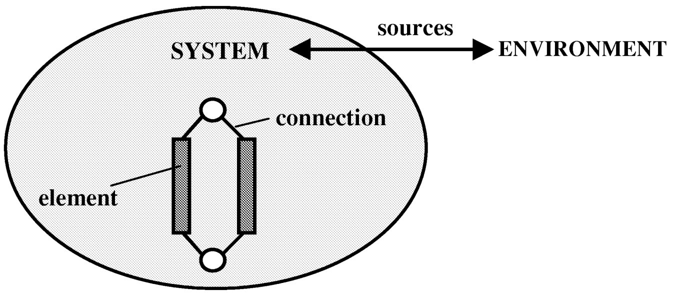


Figure-6.3: Concept of a system in manufacturing systems theory

Manufacturing system theory in the form it is presented here is due to the work of the Scandinavian School of Systems Theory for the past 30 years [Bjørke, 1995]. For detailed study of manufacturing system theory, the standard textbook used for graduate study by Professor Øyvind Bjørke should be referred [Bjørke, 1995]. By using manufacturing system theory, not only physical systems but logical systems too can be modeled and analyzed. Also, modeling approach by manufacturing system theory seems to be more general, systematic, unified and effective than the other conventional methods [Wang, 1995].

As shown in figure-6.3, a system consists of three fundamental components, namely *elements*, *connections*, and *sources*. The elements carry all the physical properties of the system. Elements are the building blocks of the physical system. For example, in an electrical LRC circuit, inductors, resistors, and capacitors are the elements; the property of a resistor is its admittance, whereas in manufacturing- a machine element's property could be its processing time, ratio between the number of input items and output items, scrap percentage etc.

When there is no connection between the elements, the set of isolated elements is called *the primitive system*. The connections reflect how the elements in the primitive system influence each other and it represents the structure of the system. The set of connected elements is called *the connected system*.

Finally, the sources reflect the influence between the total system and the environment. Sources are the environment's influence on the system; in an electrical circuit, sources are current sources or voltage sources; in production planning, demand of products is a typical source.

The Formulation Methodology in manufacturing systems theory
The objective of the new approach based on manufacturing systems theory is to offer a strategy by which the behavior of complex systems could be determined from the known behavior of its individual elements.

As stated above, a system in manufacturing systems theory is made of 1) Elements: systems parts, 2) Connections: system structure, and 3) Sources: external influence on the system. The mathematical formulation approach by manufacturing systems theory can be summarized as follow [Hussein, 1997]:

***Phase 1**: identifying the primitive system*

1. Break up the system into its basic parts, i.e. elements; this group of isolated elements is called "the primitive system".
2. Set up the governing equation of each element independent of other elements, by that, the variables in the individual elements are isolated.
3. Concurrently, by the process of measurement, an abstract model of the whole system defining the topological structure of the whole system is created.

***Phase 2**: making the connected system*

By means of the topological structure, the variables in the individual elements are connected together. That is to set up the governing equations of the whole system, or "the connected system".

***Phase 3**: applying the sources, and solving the connected system*

To apply sources and solve the governing equations of the connected system; solving the system refers to determining the behavior of the system under the applied sources.

These three phases are the guidelines in the modeling of supply chain for performance measurement, or for performance evaluation of a potential supplier.

6.3.2 Petri Net

From the previous subsection, we know that the model of supply chain that is developed by our approach has two different levels of detail, and the higher-level model that shows the flow of material between collaborating enterprises is obtained by manufacturing systems theory. It is also stated above that the lower-level model which represents the details of individual enterprises is a Petri net model.

Why Petri net?
We chose Petri nets to represent the lower-level models for obvious pro- Petri net reasons like [Silva et al, 1998],

- *Generality*, or descriptive power, with reference to production systems or any discrete event systems
- *Adequacy for dealing with real systems*. Modeling by Petri nets enables analysis to gain insight into basic structure and relevant parameters of a production system, their influences, the causes for problems like bottle neck, starvation and deadlock (stoppage due to lack of material), boundedness (inventory, surplus WIP), etc.
- *Ease of use*; doing simulations by graphical (or visual) simulators and by linear algebraic equations based simulators are easy.

6.4 MODELING SUPPLY CHAIN WITH PETRI NETS

In this section, we shall explore the use of Petri nets for modeling supply chain (lower-level model) . We start modeling supply chain with a classical approach, which we call the integrated approach; we see that the integrated approach is not suitable for modeling large systems like supply chain in agile virtual enterprise in-order to measure the performance of it, because as the number of collaborating enterprises increase, the Petri net model tends to 'explode'. Later we shall try the usage of modular modeling approach; in this modular modeling approach, we first model collaborating enterprises as individual modules and then connect these modules to the form the complete model of a supply chain. Though modular modeling approach is better than the integrated approach because the complete model obtained is less complicated than that of integrated approach, still the complete model obtained is too large for computations. Therefore, we try to reduce the size of each modules before connecting them together; literature offer some reduction theorems for this purpose. After reducing the sizes of the modules (by applying the reduction theorems), connecting the modules together gives the model of a supply chain that is of much less size, therefore better for computation. However, though the model is of minimal size thus good for computations, it is still not useful for performance measurement of supply chain in agile virtual enterprises. This is because, the modular

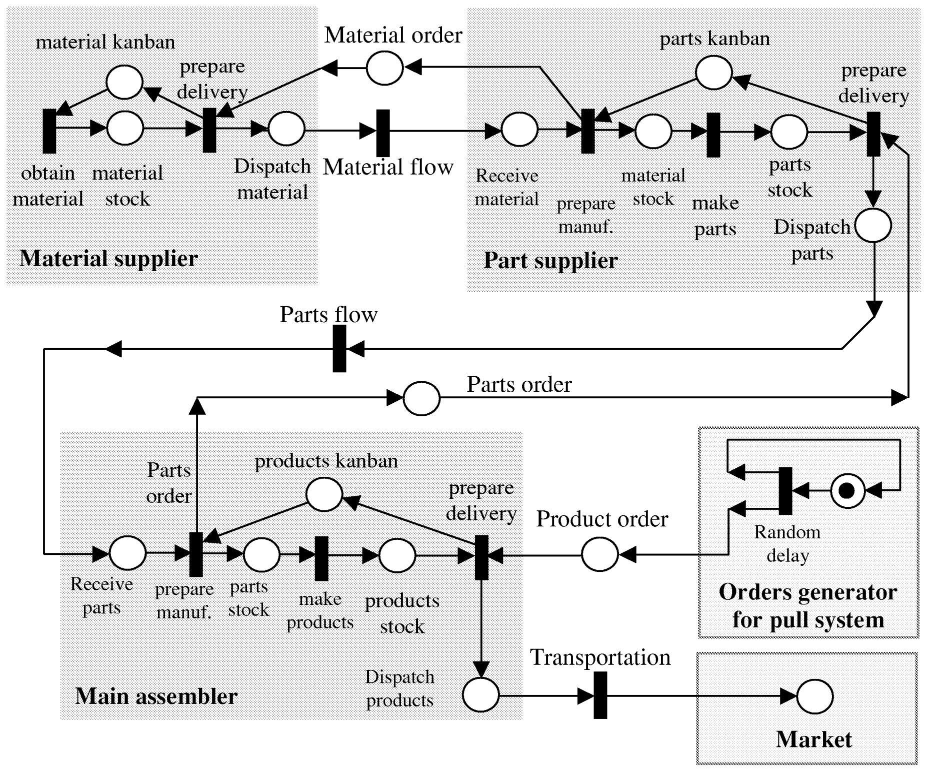


Figure-6.4: A simple PN model of a pull system, consisting of a material supplier, part supplier and an assembler

approach does not support the dynamic nature of the supply chain in agile virtual enterprises where collaborating enterprises change often. Finally we see that in the new approach, we get the lower-level model (Petri net model) of a supply chain by representing each enterprise by just a single transition-place pair and each transportation between any two enterprises is also represented by a transition-place pair. Thus this new approach guarantees that the complete model of a supply chain will be of minimal size, therefore allow fast simulation. Also, the dynamic collaboration of the virtual enterprise is taken care of the higher-level model of the supply chain.

6.4.1 Modeling supply chain: the integrated approach

The integrated model of a supply chain can be obtained by two ways: The first way is the reduction of Petri nets model for a complete supply chain; A Petri net model of a supply chain can be built from the first principles of conditions and events of production systems in agile virtual enterprise (see figure-6.4). Since the Petri net model will be huge, it can be reduced according to some proposed reduction rules while preserving properties. The main disadvantage of this approach lies in the difficulty of finding the reducible sub-Petri nets [Savi and Xie, 1992].

The second way is to build an acceptable Petri net model by starting with a simplistic model and then adding more and more details to it so that the desired properties are guaranteed without analysis. The basic idea is to build the desired properties in the model instead of checking properties after modeling the system; thus the modeling approach is neither easy nor suited for automation [Savi and Xie 1992].

Figure-6.4 (adapted from [Bonney et. al, 1999]) shows model of a supply chain build after integrated approach. Firgure-6.4 contains only a material supplier, a part supplier and an assembler, which is far too simple than any real-life supply chain, which contains probably hundreds of collaborating enterprises. Therefore, it is clear that it will be very difficult to build and analyze the complete model of any real-life supply chain by the integrated approach, as the model will contain too many nodes.

The integrated modeling approach does not offer flexibility to support the dynamic collaboration (removal and addition of collaborating enterprises) that is characteristic of agile virtual enterprises. This is because, to delete an existing enterprise (or to add a new enterprise) each single place and transitions that represent the internals of an the enterprise has to deleted one by one (or added one by one); there is no single operation to do this. Also, after the removals of defunct places and transitions, the existing places and transitions must be rewired again.

A better approach is to model each participating enterprise as a module first, and then connect these modules together to form a virtual enterprise without loosing any properties due to connection. This approach is called the modular modeling approach, which is explained in the following subsection.

6.4.2 Modeling supply chain: the modular approach

This approach is well suited to model systems composed of independent and easily identifiable subsystems, as in the case of agile virtual enterprise. The modular approach consists of mainly, modeling individual enterprises as modules, with *event graphs*, and connecting the modules together through their input/output ports. The modular modeling approach is a five step process, the first step being identification of the independent subsystems in the overall system.

After identifying the independent subsystems, then the second step is to identify the input/output ports (i/o ports) of the subsystems. It is through the i/o ports, the subsystems are going to be connected to obtain the complete model. Figure-6.5 shows input and output ports of enterprises participating in an virtual enterprise environment.

The third step is to model the individual subsystems with Petri nets; these models are called *modules*. Figure-6.6 shows a module for a part supplier, modeled using event graph. It must be made sure that the modules representing enterprises must be event graphs. Event graphs are Petri nets where all the places have exactly one input transition and one output transition [Cassandras and Lafortune, 1999; Savi and Xie, 1992]. The restriction that the modules are to be event graph is imposed so that the modules can be reduced to a minimal size (reduction in internal transitions and places) using *reduction theorems* discussed in [Savi and Xie, 1992].

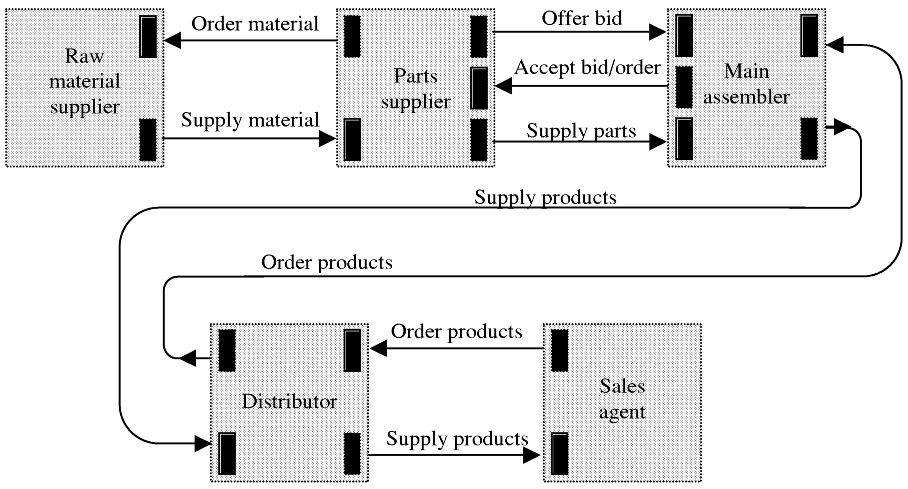
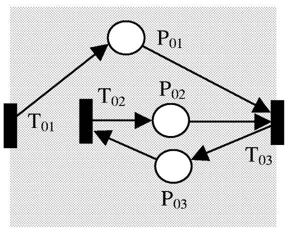
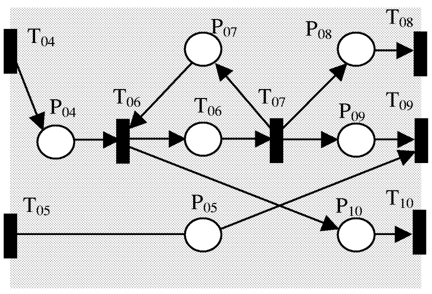


Figure-6.5: Connecting modules through i/o ports in modular modeling approach

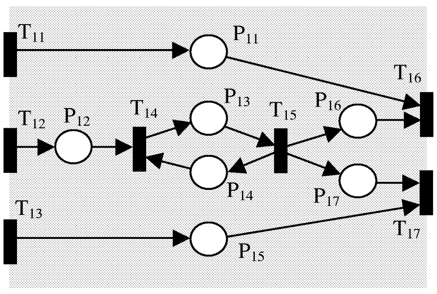
The fourth step is to reduce the size of the modules; [Harhakalis et. at., 1991; Savi and Xie, 1992] shows a reduction method that is suitable for reducing modules that contain event graphs surrounded by i/o ports (transitions). According to the reduction theorem by [Savi and Xie, 1992], each module can be replaced by an equivalent event model called minimal representation; the minimal representation has the same set of input and output transitions as the initial module, but has fewer internal places and do not has any internal transitions. Therefore, the removal of internal transitions and reduction in internal places greatly reduces the size of the connected system, thus simplifies the model analysis. The reader is referred to [Harhakalis et. at., 1991; Savi and Xie, 1992] for extensive information on reduction theorem, such as the mathematical proofs, properties like liveness, boundedness, reachability, etc.



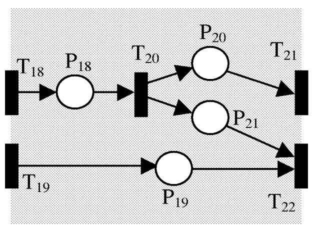
Raw material supplier



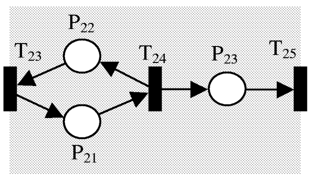
Parts supplier



The main assembler



A distributor



A sales agent

Figure-6.6: Petri net models of different kinds of enterprises

The fifth and final step is to connect modules for different enterprises together to form a complete model of the supply chain. When connecting the modules through their i/o ports, the ports are replaced by transitions. Since the i/o ports are transitions, the connecting line (arc) should contain a place.

In figure-6.6, different types of enterprises are modeled as modules. It must be remembered that modules representing enterprises must be event graphs, where all the places have exactly one input transition and one output transition. The description of places and transitions of the modules shown in figure-6.6 is given in table-6.1. It is obvious that the Petri net models for the different kinds of enterprises are much simplified diagrams, but since we are more interested in creating a complete model by connecting modules together, to get the overall performance, let us keep the modules simple.

A raw material supplier			
T_{01}	Receive order from part supplier/assembler	P_{01}	Orders for material received.
T_{02}	Production of raw material.	P_{02}	Material ready for shipment.
T_{03}	Ship material to part manufacturer	P_{03}	Volume of material to produce.
A part supplier			
T_{04}	Material arrive (from mat. supplier)	P_{04}	Unloaded material in queue
T_{05}	Orders received from assembler	P_{05}	Received orders for parts
T_{06}	Production calculations	P_{06}	Ready for production of parts
T_{07}	Manufacture of parts	P_{07}	Monitor of quantity produced
T_{08}	Order raw materials	P_{08}	Monitor of materials used
T_{09}	Ship parts to main assembler	P_{09}	Parts ready for shipment
T_{10}	Send bid to main assembler	P_{10}	Bid for parts (to assembler)
Main assembler			
T_{11}	Bids for parts and raw materials	P_{11}	Received bids from suppliers
T_{12}	Parts and raw materials arrive	P_{12}	Unloaded parts in queue
T_{13}	Orders received	P_{13}	Ready for manufacture
T_{14}	Production calculations	P_{14}	Monitor of quantity produced
T_{15}	Manufacture of products	P_{15}	Received orders for products
T_{16}	Order parts and raw materials	P_{16}	Monitor of parts/materials used
T_{17}	Ship products to distributors	P_{17}	Products ready for shipment
A distributor			
T_{18}	Goods arrive from main assembler	P_{18}	Stock of goods
T_{19}	Receive orders for goods from sales agent	P_{19}	Received orders for goods
T_{20}	Supply and distribution calculation	P_{20}	Monitor of free capacity
T_{21}	Order goods from main assembler	P_{21}	Goods ready for shipment
T_{22}	Ship goods to sales agents		
A sales agent			
T_{23}	Goods arrive from distributor	P_{21}	Stock of goods
T_{24}	Sales	P_{22}	Monitor of free capacity
T_{25}	Order goods from distributor	P_{23}	Monitor of sales

Table-6.1: Description of the modules for different kinds of enterprises

The module for a raw material supplier in figure-6.6 has three transitions, one input transition (marked T_{01}), one output transition (T_{03}), and an internal transition (T_{02}); it is clear that T_{01} refers to the input port and T_{03} refers to its output port. Also, the module has three internal places P_{01} - P_{03}. In the Petri net module for a sales agent is also shown in figure-6.6, sale of goods is represented by the transition T_{24} (see table-6.1 too). Though sales means goods leaving agent, therefore an output activity, it is shown as an internal transition because sales is not coupled directly with a distributor.

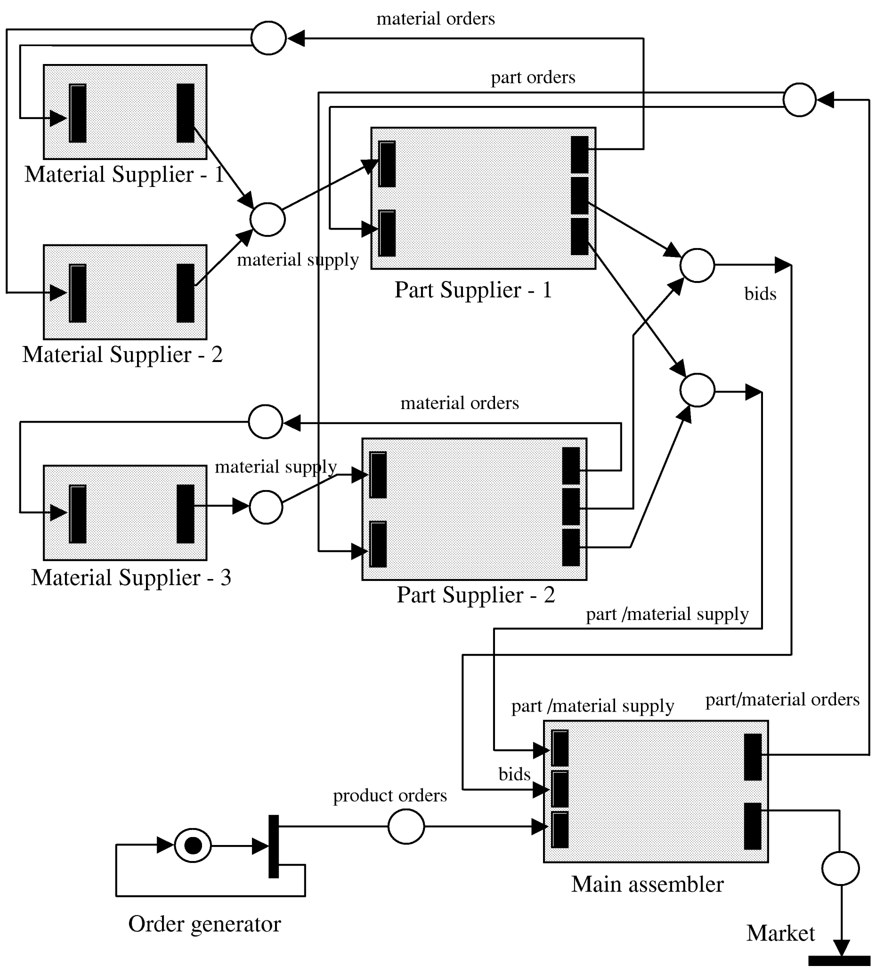


Figure-6.7: Model of a trivial supply chain obtained by connecting modules of material suppliers, part suppliers and the main assembler.

The supply chain model of a virtual enterprise is obtained by connecting the modules for different kinds of enterprises through their i/o ports. Since the i/o ports are transitions, the connecting line (arc) should contain a place, see figure-6.7. Please note that in figure-6.7, only the modules are event graphs. The places that connect the modules together are part of ordinary Petri nets, thus can have many inputs and outputs.

Figure-6.7 shows a small system consisting of just six enterprise, the main assembler (there will just one main assembler in an agile virtual enterprise, as our view of agile virtual enterprise is a 'main assembler-centered' view) two parts suppliers (first-tier suppliers) and three raw material suppliers (second-tier suppliers). Even for this small system model, there are many places (39 places totally) and transitions (30 transitions); thus, analyzing a real-world agile virtual enterprise consisting hundreds of supply and distribution enterprises will be difficult. There must be a way of reducing individual modules to a minimal size while preserving their properties. The next subsection is about reduction of modules.

6.4.3 Reduction of modules

By reduction theorem, simple event graph module called minimal representations replace more complex modules for different kinds of enterprises. By reduction, the liveness and boundedness properties of the original module are preserved. The procedure of obtaining a minimal representation is given in [Savi and Xie, 1992].

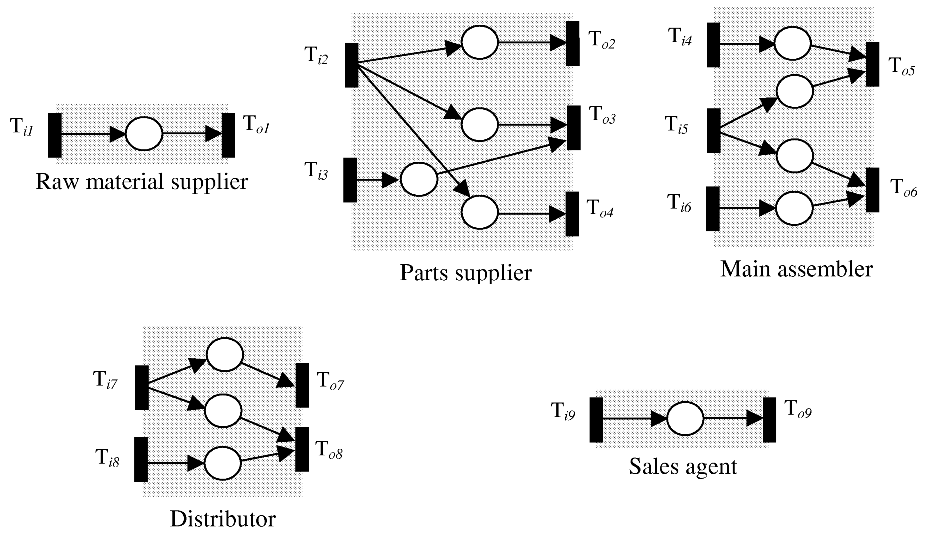


Figure-6.8: Minimal representations for different types of enterprises

The minimal representations for the different types of enterprises are shown in figure-6.8. If we replace the original modules in the model shown in figure-6.7 by the reduced modules, then the connected model will have 21 transitions and 24 places totally. Thus, the reduction is the total number of nodes (transitions 30% and places 38%) is significant; If we modeled the enterprises more realistically (more complex), then the difference due to reduction will be very significant.

It is obvious that the complete model of a supply chain obtained by the modular approach (connecting minimal representation of participating enterprises) will be of much smaller size than the model obtained by integrated approach. Therefore, it will be easy to analyze the model obtained by the modular approach , using any standard mathematical analysis techniques for Petri nets. Material flow amounts, arrival times, starvation due to lack of material (liveness) or over production and stocking and WIP can be calculated using standard techniques for Petri nets. By assigning realistic data, these simulations could be carried-out.

It must be noted that the modular approach too is not quite suitable for modeling supply chain in agile virtual environment where the collaborating enterprises dynamically change. Again, this is because of the fact that to delete an enterprise from collaboration, the module representing the enterprise has to be deleted manually, and then new connections has to be made between the remaining places and transitions. To add a new enterprise into collaboration, similar changes must be made manually.

6.4.4 Further reduction of modules

After reducing the modules, the minimal representations have few transitions and places; for example, in figure-6.8, the minimal representation of a part supplier has 5 transitions (2 input transitions and 3 output transitions) and 4 places (internal places). In our approach discussed in the next section, we trim down the sizes of the modules even further; the modules for any enterprise has just two nodes - a transition and a place. The transition represents the overall production at the enterprise and the place represents the output buffer for products to be pulled out by the purchasing enterprise. The transportation between any two enterprises is also replaced by a transition-place pair. The transition of a transportation represents the transport of material between the enterprises involved, and the place represents the input buffer for material for the purchasing enterprise. The reason for reducing a module for an enterprise or a transportation into just a transition-place pair is that, we are interested in the overall performance of the collaboration and not on the internal performance of an enterprise. These ideas will become apparent when we go through detailed modeling later.

6.5 REALIZING PERFORMANCE EVALUATION ENGINE

Before explaining the different stages of developing (modeling, simulation and implementation) the performance evaluation engine, let us look into the requirements of it. The performance evaluation engine should offer:

1. Performance measurement with key performance indicators cost, quantity, and time,
2. Performance evaluation of potential suppliers (adding new partners to collaboration),
3. Fast computation, because the engine will be used online for real-time application, and
4. Easy to use and simple user interface.

Realization of the performance evaluation engine is done in three stages. Figure-6.9 shows the different stages. The first stage is the modeling stage where we use manufacturing systems theory approach to establish a higher-level model of the real-life agile virtual enterprise. This higher-level model is also called the *connected system*. At the modeling stage, the tool AgileSIM is used to establish the connected system (AgileSIM is created by the author and runs on the MATLAB system).

The second stage is the simulation stage. After establishing a connected system, the modeling tool AgileSIM automatically generates the lower-level Petri net model of the connected system. The tool PenSIM (Petri Net Simulator, an integral part of AgileSIM) can be used to carry-out simulations on the Petri net model, and this is the reason for calling this stage - the simulation stage. Thus the *dynamic collaboration* of the virtual enterprise is expressed by the connected system, while the *dynamic activities* of the virtual enterprise is represented by the places and transitions of the Petri net model.

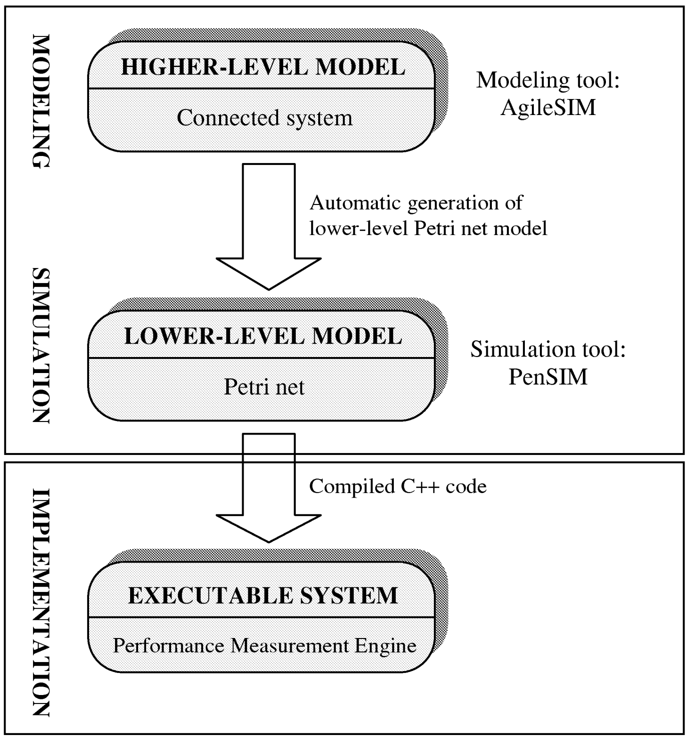


Figure-6.9: Different stages of developing the performance evaluation engine

The third stage is the implementation stage. At this stage, the Petri net model is converted to C++ programming language code and compiled into a executable system - the performance evaluation engine. As shown in figure-6.1, this executable system will co-operate with at least two other executable systems, the inference engine and the data collection system.

In the following sections 6.6, 6.7, and 6.8 the different stages (modeling, simulation and implementation, respectively) are explained in detail.

6.6 MODELING STAGE OF PERFORMANCE EVALUATION ENGINE

The modeling stage is the initial stage of developing the performance evaluation engine (see figure-6.9). The virtual enterprise is modeled at this stage with the tool AgileSIM. The approach used for modeling at this stage is based on manufacturing systems theory. Basics of manufacturing systems theory is given in subsection 6.3.1.

Considering supply chain in agile virtual enterprise as a system, there exist two types of elements; the collaborating enterprises is the first type of elements and the transporting agents between any two enterprises is the second type of elements. It is very important to note that in virtual enterprises where the enterprises may be geographically spread all around the world, the transportation (or transporting agents) between the collaborating enterprises are as equally important as the collaborating enterprises themselves. Thus, a group of isolated enterprises and transporting agents make-up the primitive system. An enterprise has *properties* like production costs per unit, production time, etc., whereas a transportation has properties like supplying enterprise, purchasing enterprise (that is transport from whom, to whom), transportation delay, transportation costs, etc.

A *network* is a set of transportation, thus connecting together enterprises in the primitive system to establish the connected system. Figure-6.10 shows a network of a single transportation, connecting three primitive elements (two enterprises and a transporting agent between the enterprises) together forming a connected system.

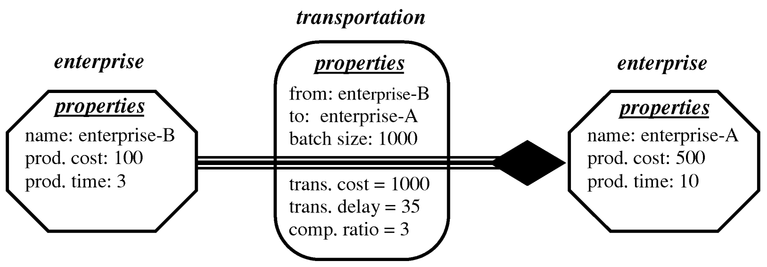


Figure-6.10: A connected system, consisting of three primitive elements

Careful reader may sense some subtle difference between our use of manufacturing systems theory and the theory described in [Bjørke, 1995]; Let us explain: In manufacturing systems theory, the connection between any primitive elements is *passive* meaning the connecting line has no values. Only the primitive elements have values or properties. By this argument, the transportation is also a kind of a connection, connecting two enterprises, thus should not have any values. But considering the nature of virtual enterprise, we may then say, that along the connection between two enterprises (primitive elements) there is *point* such that it has a cost value which is equal to the cost difference between a unit of material departing the supplying enterprises, and arriving the purchasing enterprise (see figure-6.10). This point also has a time delay which is the time difference between the time of arrival of material at the purchasing enterprise and the time of departure from the supplying enterprise. And finally, this point is called the transportation agent. What we are trying to say is that the transportation between any two enterprises is not trivial as the copper wire connecting two resistors an electric circuit; the copper wire does not has any impedance compared to the resistors, whereas transporting agent does influence the material flow between any two enterprises, primarily with reference to cost and time.

6.6.1 Obtaining higher-level model using AgileSIM

Obtaining a higher-level model of virtual enterprise consisting many collaborating enterprises connected together by a network of transportation, is a three phase process as described in subsection 6.3.1.

As an example, let us use AgileSIM to get the higher-level model of the simple system shown in figure-6.10.

Phase 1: identifying the primitive system

Let us define the primitive elements shown in figure-6.10, one by one.
To define enterprise-B,

```
> B = enterprise(prod_time, prod_cost, 'B');
```

where the first parameter is the time taken for overall production at the enterprise-A, and the second parameter is the cost of production per unit. The last parameter is string of text used as a label or name for the enterprise. Similarly, enterprise-A could be defined.

To define the primitive element the transportation (between 'B' and 'A'),

```
> BA = transport(c_ratio, batch_size, t_delay, t_cost, B, A, 'BA');
```

Here, the first parameter is the component ratio (CR). Component ratio means how many units of products flowing through this transportation is needed to turn one unit of product at the purchasing enterprise. The second parameter is the batch size. This means, the material flow amount through this transportation is a whole multiple of the batch size. The third parameter is the transportation delay. The fourth parameter is the transportation cost per batch. The fifth and sixth parameters are the supplying enterprise and the purchasing enterprise for this transportation. The last parameter is the label or name of this transportation.

Phase 2: making the connected system

There is only one transportation in figure-6.10. Therefore the collaboration has only one transportation.

```
> VE = collaboration(BA, 'a simple Virtual Enterprise')
```

By executing the statement above, the connected system VE (VE stands for virtual enterprise) is established. This connected system is the higher-level model.

Finally,

Phase 3: applying the sources, and solving the connected system

Let say that the enterprise-A is to produce 30 units of products. Thus the source for this system is 30 units in enterprise-A.

```
> source1=sources(A, 30);
```

To create the lower-level model, the source is applied to the connected system:

```
> petri_model = convert (VE, source1);
```

By executing the above program line, AgileSIM generates the lower-level model - the Petri net model. The Petri net model can be simulated using the Petri net simulator PenSIM.

6.7 LOWER-LEVEL MODEL: THE PETRI NET

Once by system specification a connected system (higher-level model) is made, the lower-level Petri net model generated by AgileSIM is based on timed colored Petri net. In the Petri net, the places does not simply hold numerals (or tokens) as in the case of ordinary Petri nets, but data like the inventory levels, accumulated costs etc (colored tokens). Also, the transition does not fire just because the input place has enough tokens, but fires if the logical conditions attached to it are satisfied. Before formally defining the colored Petri net used in AgileSIM/PenSIM, let us first define the ordinary Petri net:

6.7.1 Ordinary Petri net

■ Definition 6.1 (ordinary Petri net):

A Petri net is a four-tuple

$$(P, T, A, x_0)$$

Where,

P is the set of places (representing conditions or number of parts), $P = [p_1, p_2, \ldots, p_n]$

T is the set of transitions (corresponds to events), $T = [t_1, t_2, \ldots, t_m]$

$A \subseteq (P \times T) \cup (T \times P)$ is a set of arcs from places to transitions and from transitions to places, and

$x = [x(p_1), x(p_2), \ldots, x(p_{n1})] \in \mathrm{N}^n$ is the row vector of markings (tokens) on the set of places, x_0 is the initial marking. ■

Let *I(t$_j$)* represent the *set of input places* to transition t_j and *O(t$_j$)* represent the *set of output places* to transition t_j. Then,

$$I(t_j) = \{p_i \in P : (p_i, t_j) \in A\}, \qquad O(t_j) = \{p_i \in P : (p_i, t_j) \in A\}$$

Similarly, let *I(p$_i$)* and *O(p$_i$)* be the set of input transitions and the set of output transitions to the place p_i respectively.

The weight of an arc from p_i to t_j: if there are k arcs directed from transition t_j to place p_i $p_i \in O(t_j)$ then the weight of the arc $w(t_j, p_i) = k$.

Having defined the ordinary Petri net, now the enabled transitions (conditions that make a transition in ordinary Petri net to fire) is defined:

■ <u>Definition 6.2 (Enabled transition in ordinary Petri net)</u> [Cassandras and LaFortune, 1999]:
A transition $t_j \in T$ in an ordinary Petri net is said to be *enabled* if

$$x(p_i) \geq w(p_i, t_j) \text{ for all } p_i \in I(t_j). \quad ■$$

In other words, transition t_j in the ordinary Petri net is enabled when the number of tokens in p_i is at least as large as the weight of the arc connecting p_i to t_j. When a transition fires, the vector of markings change:

■ <u>Definition 6.3 (Firing in ordinary Petri net)</u>:
A transition $t_j \in T$ fires in an ordinary Petri net, if and only if $x(p_i) \geq w(p_i, t_j)$ for all $p_i \in I(t_j)$, changes occur in markings

$$x'(p_i) = x(p_i) - w(p_i, t_j) + w(t_j, p_i), \quad i = 1, \ldots, n. \quad ■$$

With these definitions, now it is high time to go through the definitions for the new high-level Petri net.

6.7.2 Timed Colored Petri net

In this subsection, we present definitions for the timed colored Petri net used by the AgileSIM/PenSIM, together with the ***new firing rules***.

■ <u>Definition 6.4 (the timed colored Petri net)</u>:
The timed colored Petri net used by AgileSIM is a four-tuple

$$(P, T, A, x_0)$$

where

(P, T, A, x_0) is an ordinary Petri net

P is the set of places partitioned into subsets P_E and P_C, such that $P = P_E \cup P_C$, and

T is the set of transitions partitioned into subsets T_E and T_C , such that $T = T_E \cup T_C$.

Also,

For a transition $t_{jE} \in T_E$, there is only one input place, i.e. $\|I(t_{jE})\| = 1$, and

For a transition $t_{jC} \in T_C$, there is only one input place and only one output place. i.e. $\|I(t_{jC})\| = 1$, and $\|O(t_{jC})\| = 1$. ■

Subscripts *E* and *C* refers to enterprise and transport respectively.

■ Definition 6.5 (Firing in the timed colored Petri net):

When transition $t_j \in T$ fires in the timed colored Petri net, changes occur in markings in the following way:

When transition $t_{jE} \in T_E$ fires:

For the input place $p_{iE} \in I(t_{jE})$, $x'(p_{iE}) = 0$

For all the output places $p_{oC} \in O(t_{jE})$, $x'(p_{oC}) = x(p_{iE}) \times w(t_{jE}, p_{oC})$, $o = 1, \ldots, n.$

When transition $t_{jC} \in T_C$ fires:

For the input place $p_{iC} \in I(t_{jC})$, $x'(p_{iC}) = 0$

For the output place $p_{oE} \in O(t_{jC})$,

$$x'(p_{oE}) = m \times w(p_{iC}, t_{jC}) \triangleright x(p_{iC}), \quad m \in \aleph^+$$

which means, markings on the output place is the smallest multiple of the weight of the arc *w(p_{iC},t_{jC}),* which is greater than or equal to the markings on the input place of the transition. ■

■ Definition 6.6 (Enabled transition in the timed colored Petri net):

A transition $t_j \in T$, $T = T_e \cup T_c$ in the timed colored Petri net is said to be *enabled* if

$x(p_i) \geq 0$ for all $p_i \in I(t_j)$, and

$x'(p_o) \leq \lceil x(p_o) \rceil$ for all $p_o \in O(t_j)$, ■

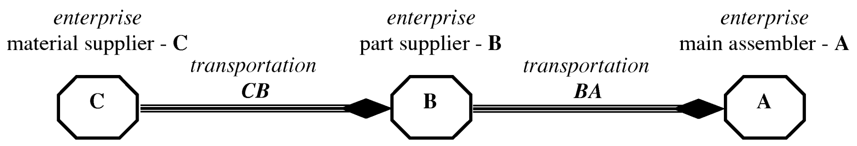
In other words, transition t_j is enabled when there is *any* number of tokens in the input place p_i and if the tokens added to the outplace by firing will not exceed the limit (maximum) attached to that output place.

Further explanation for these firing rules are given in subsection 6.8.3, after the algorithms for simulations are introduced.

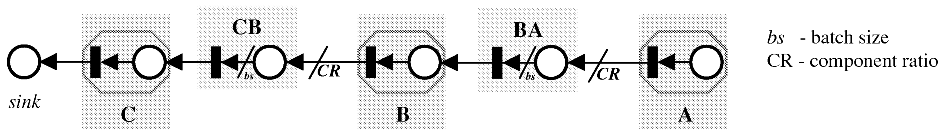
6.7.3 Generating Petri net model from connected system

When AgileSIM generates the Petri net model of the connected system, the following rules are observed:

1. Each enterprise is replaced by transition-place pair (t_E - p_E). The transition represents the (overall) production at the specific enterprise and the place represents the out-buffer for products of that enterprise. See figure-6.11.
2. Each transportation is replaced by transitions-place pair (t_C - p_C). The transition is connected to the supplying enterprise ('from' enterprise), and the place is connected to the purchasing enterprise ('to' enterprise). The weight of the arc between the transition and the place is the batch size.
3. The direction of flow in Petri net model is opposite to that of the connected system model. Since the pull system for material flow control is assumed, Petri net model depicts the order flow (or information flow) which should be in opposite direction to connected system model which shows the material flow direction.
4. The weight of the arcs between the places of the transportation (p_C) and the purchasing enterprise transition (t_E) is the *component ratio*. Component ratio means how many units of parts flowing through a connection is needed to make a unit of product at the purchasing enterprise.
5. The raw material suppliers or any other enterprises that are not connected to any supplying enterprises in the connected system become *sinks* in the Petri net model. Therefore, extra places are attached to them to accumulate the tokens.



6.11a: A simple connected system



6.11b: Petri net model for the connected system in 11a.

Figure-6.11: Generating Petri net model from a connected system

Figure-6.11 shows the rules for generating Petri net model (lower-level model) from the connected system (higher-level model) pictorially. Though the Petri net model is shown pictorially in figure-6.11b, AgileSIM creates Petri net model in a matrix form called *the incidence matrix*. The incidence matrix representing a Petri net is actually a matrix of input places and output places of the transitions in the Petri net model. Figure-6.12 shows the incidence matrix for the Petri net model shown in figure-6.11b.

6.8 SIMULATIONS WITH PETRI NET MODEL

The use of Petri net for lower-level modeling makes sure that the simulations (time, amount and cost calculations) will be done in linear time. The methodology that is used to manipulate the lower-level Petri net model in calculations makes sure that the calculations will done in linear time `O(n)` where `n` is the number of transportation. This requirement on linear time is a necessity, otherwise because of the size of the system model (large number of enterprises and transportation), non-linear simulation time will make the implementation inappropriate for online real-time use.

Calculations are done in two traversals. In the first traversal (*'walk-through'*), the simulation follows the order-flow direction starting at the source (for example, in figure-6.11a, simulation starts at enterprise-A, the right-end of the Petri net) with demands on products as the initial markings. During this traversal, the material flow amount and time profile are calculated. In the second traversal (*'back-tracking'*), the simulation start at the sink, (in figure-6.11a, the left-end of the Petri net model, enterprise-B) following the material-flow direction. To do this, the flow direction of the Petri net model must be reversed. This reversing is done simply by swapping the input places and output places in the incidence matrix (see figure-6.12). During this second traversal, the cost calculations are done.

The most time consuming activity of this research is programming AgileSIM/PenSIM in MATLAB environment. In the following subsections, only the simplified versions of the algorithm behind AgileSIM/PenSIM is given. Interested reader is encouraged to inspect the program codes at the web site [AgileSIM, 2000].

$$incidence\_matrix = \begin{array}{c|cccccc|cccccc|} & p_A & p_B & p_C & p_{BA} & p_{BC} & p_{\sin k} & p_A & p_B & p_C & p_{BA} & p_{BC} & p_{\sin k} \\ t_A & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & CR_{BA} & 0 & 0 \\ t_B & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & CR_{CB} & 0 \\ t_C & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ t_{BA} & 0 & 0 & 0 & bs_{BA} & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ t_{CB} & 0 & 0 & 0 & 0 & bs_{CB} & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ & \multicolumn{6}{r}{input\ places} & \multicolumn{6}{r}{output\ places} \end{array}$$

Figure-6.12: Incidence matrix for the Petri net model shown in figure-6.11b.

6.8.1 Algorithm for the first traversal ('Walk-through')

In the first traversal called Walk-through[3], calculation starts at the sources (order generators), and ends at the sinks, following the order flow direction. The material flow amount and time profile are calculated in this traversal.

The algorithm for the material flow amount calculation is shown here as algorithm-6.1. During the first traversal, the time calculation are also done at the same, but not shown in algorithm-6.1 for brevity. The time profile is made in the following way:

```
if the firing transition is a $t_C$ then {
        add time equal to the transportation time
} // if $t_C$

if the firing transition is a $t_E$ then {
        add time amounting to (production_time_per_unit * units_produced)
} // if $t_E$
```

```
start at the sources;
repeat {
        if the firing transition is a $t_C$ (meaning a transport transition) then {
                remove material amount at place $p_C$
                increase material amount to whole multiple of the batch size of transportation
                deposit material amount to $p_E$, the place at the supplying enterprise
        } // if $t_C$

        if the firing transition is a $t_E$ (meaning an enterprise transition) then {
                remove material amount at place $p_E$
                for each transportation attached to the enterprise at the supply side {
                        multiply material amount by component ratio (CR)
                        deposit material amount to $p_C$, the place at the transportation agent
                } //for each transportation
        } // if $t_E$
} until at sink
```

Algorithm-6.1: Calculating material flow amounts during *"walk-through"*

6.8.2 Algorithm for the second traversal ('Back-tracking')

In the second traversal called Back-tracking[4], calculation starts at the sinks and ends at the sources, following the material flow direction. In this second traversal, the cost of product as it goes through different enterprises (productions) and transportation is calculated. The algorithm for the cost calculation is shown in algorithm-6.2.

[3] Walk-though: a perfunctory performance of a play or acting part, as in an early stage of rehearsal [Merriam-Webster's Collegiate Dictionary, Internet Ed.]

[4] Back-tracking: to retrace one's course / to go back to an earlier point in a sequence / to reverse a position [Merriam-Webster's Collegiate Dictionary, Internet Ed.]

6.8.3 Explanation for the new firing rules

Having seen the algorithm for the first and second traversals, we would like to explain why we devised some peculiar firing rules for the timed Petri net. The firing rules are formally presented in subsection 6.7.2.

The first firing rule says that when a transition (enterprise or transportation) fires, it removes all the tokens from the input place. This is alright, because when we transport (transport transition fires), we do transport the whole batch, not one by one. The firing rule is also true for enterprise transition: even if products are being produced one by one in an enterprise, when considering the overall performance, we do consider the overall production time, cost etc., assuming the whole batch of products are produced at a time.

The second firing rule says that when tokens are deposited in the outplace of an enterprise transition, number of tokens deposited is equal to weight of the arc between transition and output place multiplied by the input tokens. During the first traversal ('walk-through'), we follow the order flow. Thus to produce a n number of product at an enterprise, we need a particular raw material amounting to n multiplied by the component ratio. Component ratio is the weight of the arc that is between the enterprise transition and the output place of it.

Finally, the third firing rule says that when a transportation transition fires, the number of tokens deposited into the output place is the smallest multiple of the batch size, and this number is greater than or equal to the input tokens. Imagine that the batch size for transportation 2000, and the input tokens (representing units of products to be transported) are 3400. But the transportation can not transport 3400 units, but only a multiple of the batch size (2000). Thus the transported units are (deposited tokens in the output place) actually 4000. Since the actual number needed was 3400, the rest 600 is inventory left in the receiving enterprise.

```
start at the sinks;
repeat {
        if the firing transition is a $t_C$ then {
                cost addition per unit = (transportation cost)/(batch size)
        } // if $t_C$

        if the firing transition is a $t_E$ then {
                cost addition per unit =product cost per unit;
                for each transportation attached to the enterprise at supply side {
                        raw material cost = (component ratio * raw material cost per unit)
                        cost addition per unit = cost addition per unit + raw material cost
                  } // for each transportation
        } // if $t_E$
} until at source
```

Algorithm-6.2: Cost calculation during *"back-tracking"*

6.9 SIMULATION EXAMPLE

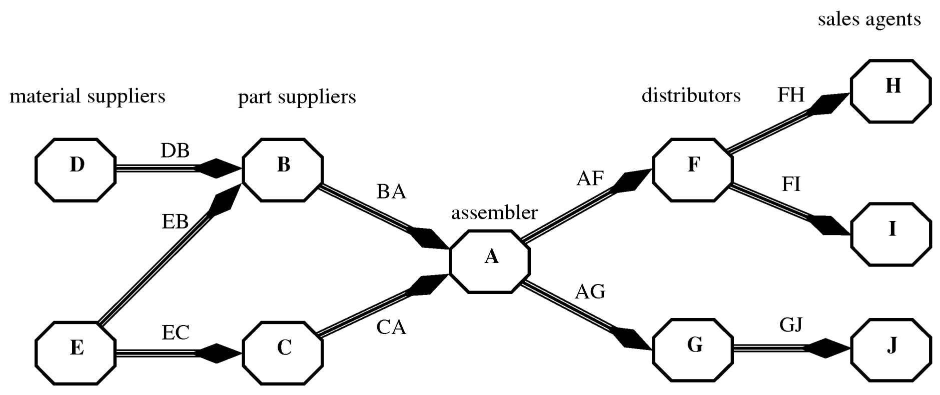


Figure-6.13: The connected system for the application example

A small connected system consisting of 10 enterprises (A-B-C-D-E-F-G-H-I-J), and linked together by 10 connections (BA-CA-DB-EB-EC-AF-AG-FH-FI-GJ) is shown in figure-6.13; the properties of the enterprises and the transportation are given in table-6.2 and in table-6.3 respectively. Suppose some customers order a number of products at the sales agent 'H'. The first part of this example is to compute the time taken (time profile) to satisfy the demand, the inventory left in the participating enterprises and how the costs are added to the product as the product is developed through the supply chain. The second part of this example is about evaluating a potential supplier/distributor.

Let us assume that the demand at sales agent 'H' is 30. This will be the source of the connected model. Now that all the necessary data are given for the system specification (table-6.2/ table-6.3), given below is the mathematical formulation approach for problem solving:

6.9.1 Performance measurement of supply chain

Phase 1: identifying the primitive system

The primitive systems consists of 10 enterprises and 10 transportation, the properties are given in table-6.2 and in tabel-6.3 respectively. Using AgileSIM these primitive elements are defined first. For example, to define enterprise-A (the main assembler) using the properties given in table-6.2,

```
> A = enterprise(0.3, 250, 1000, 'A');
```

Similarly, the other enterprises are defined.

Table-6.2: Properties of the enterprises

Enterprise	*Properties*		
	Production Cost (per unit)	*Production Time (hours)*	*Max. inventory (in units)*
A	250	0.3	1000
B	50	0.1	2000
C	25	0.1	3000
D	20	0.05	4000
E	25	0.04	5000
F	170	0.1	1000
G	175	0.1	1000
H	285	0.1	100
I	290	0.1	100
J	290	0.1	100

Table-6.3: Properties of the transportation

Transport-ation	*Properties*					
	From	*To*	*Batch Size*	*Trans. Cost per batch*	*Transport Time hours*	*Component Ratio*
BA	B	A	1000	100	48	2
CA	C	A	1500	100	24	3
DB	D	B	1500	200	48	3
EB	E	B	1000	140	72	2
EC	E	C	2000	140	72	1
AF	A	F	100	250	168	1
AG	A	G	100	250	168	1
FH	F	H	10	50	48	1
FI	F	I	100	50	48	1
GJ	G	J	100	45	24	1

To define transportation-BA (between enterprise-B and enterprise-A) using the properties given in table-6.3,

```
> BA = transport(2, 1000, 48, 100, B, A, 'BA');
```

Similarly, the nine other transportation are defined.

Phase 2: making the connected system

The collaborating enterprises are grouped together to form the connected system. This is done in AgileSIM by declaring the network of transportation:

```
> VE1 = collaboration(BA,CA,DB,EB,ED,AF,AG,FI,FH,GJ, ' Virtual Enterprise-1');
```

Phase 3: applying the sources, and solving the connected system

There is only one source for this system, which is 30 units in enterprise-H. Thus,

```
> source1=sources(H, 30);
```

Finally, to get the Petri net model,

```
> Petri_model1 = convert (VE1, source1);
```

Calculations

Petri net model can be simulated using the Petri net simulator- PenSIM.

```
> [time_profile, material_flow, cost_calculation] = pensim(petri_model1);
```

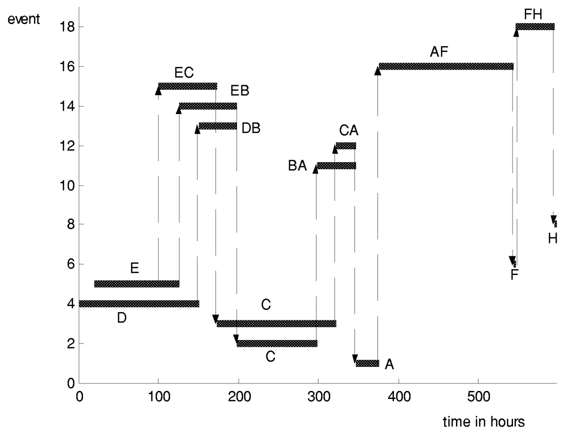


Figure-6.14: Time profile of the product flowing through the supply chain

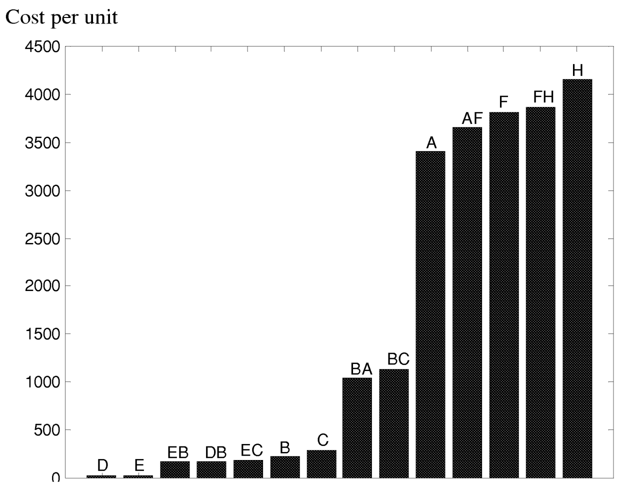


Figure-6.15: Cost of a unit product as it flow through the supply chain

The results of the simulations, time series ('time_profile'), flow amounts ('material_flow') and incurred costs ('cost_calculation') are in matrix form, therefore can be used by any mathematical analysis packages. Plotting these matrices with MATLAB system gives the figures-6.14 to 6-16. Figure-6.14 shows the time profile. It shows the time taken for production at different enterprises ('A' to 'H') and the time taken due to transportation ('BA' to 'FH') between enterprises. From this figure, among other things- it is easy to identify the most time consuming task.

Figure-6.15 shows how the cost of a product increases as it flows through different enterprises and transportation, starting from the material suppliers 'D' and 'E', through the main assembler 'A', to the final destination- the sales agent 'H'. From this figure, it is easy to identify the most expensive/least expensive production node and transportation, for example. Figure-6.16 shows the inventory left at different enterprises after production. From this figure, it is easy to identify the unnecessary inventory accumulation along the supply chain.

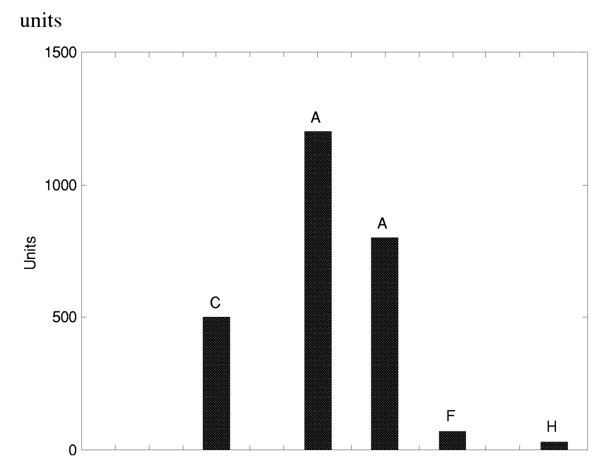


Figure-6.16: Inventory levels left at enterprises

6.9.2 Performance Evaluation of a potential supplier

An important criteria for AgileSIM is that, there is support for dynamic selection of collaborating enterprises. Suppose a new enterprise, lets call it enterprise-X, a part supplier offers competitive prices, and the decision makers wants to see whether replacing enterprise 'C' by 'X' will improve the overall performance. Using AgileSIM, this kind of studies can be done efficiently, using a few program codes:

Phase 1: identifying the primitive system

First we want to remove enterprise C from the primitive system.

```
> VE2 = remove_element(C, VE1) %new VE obtained by removing C from VE1
```

Now the new enterprise-X is defined so that it can be added to the primitive system:

```
> X = enterprise(0.1, 35, 'X'); %prod_time=0.1 hour, prod_cost=35,
```

Phase 2: making the connected system

When an enterprise is removed from primitive system, all the connections it has with the rest of the connected system are also removed automatically. We need only to define the transportation exist between the new enterprise and the rest of the system. E.g.:

```
> XA = transport(3,1000,24, 90, X, A, 'XA'); %defining new transportation XA
```

And finally, to form the (new) connected system, the new collaboration is defined:

```
> AVE2=add_collaboration(XA,VE2);%new connected system reflecting new collaboration
```

Phase 3: applying the sources, and solving the connected system

AgileSIM function 'convert' is executed again to generate the new Petri net model. After this, the Petri net model can be used to carry out simulations.

```
> petri_model2=convert (AVE2, source1); %no change in demand, thus source1
```

The simulation results due to the new collaboration with enterprise-X can now be compared with the results obtained from older collaboration, to determine whether replacing 'C' with 'X' will improve the overall performance of the supply chain. Decision can be made based on the policies like *fastest response time* (by comparing figure-6.14 with the time profile of the new collaboration), *cheapest price* (by comparing figure-6.15 with the costs calculations from the new collaboration), or *lowest inventory levels* (comparing figure-6.16 with the inventory levels from the new collaboration). Since the simulation results from the old and new collaborations are in matrix form, any mathematical packages can be used for comparison.

Figure-6.17 compares the time profiles of the two collaborations. After receiving the customer order at the sales agent 'H', total time taken to furnish finished products at 'H' is about the same in both collaborations. This means, the enterprise-C and enterprise-X have negligible influence on the whole production time.

Figure-6.18 shows how the costs (due to production/transportation) are added to the product development. By the new collaboration, the total cost of a unit product at the sales agent 'H' is lower (3660 monetary units) compared with older collaboration (4155 monetary units).

Finally, figure-6.19 compares the inventory levels left at different enterprise due to production. In comparison, the new collaboration fares well, as the inventory levels at 'C' is completely eliminated (because 'C' is no longer a collaborating enterprise), and the inventory levels at the main assembler 'A' is reduced; 'A' has two types of inventory, therefore 'A' is given two columns in figure-6.19. The inventory levels at 'F' remains same for both collaborations. In figure-6.19, the inventory level at sales agent 'H' indicates the customer orders.

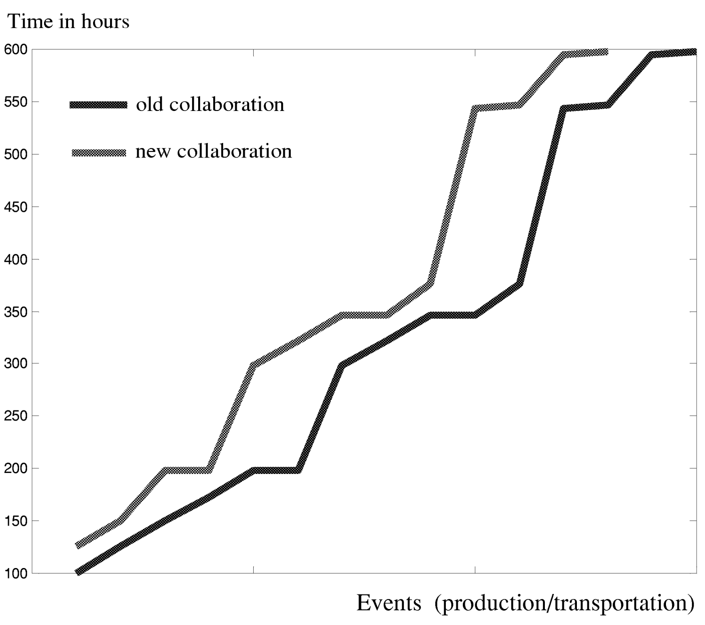


Figure-6.17: Comparison of time profiles of collaboration

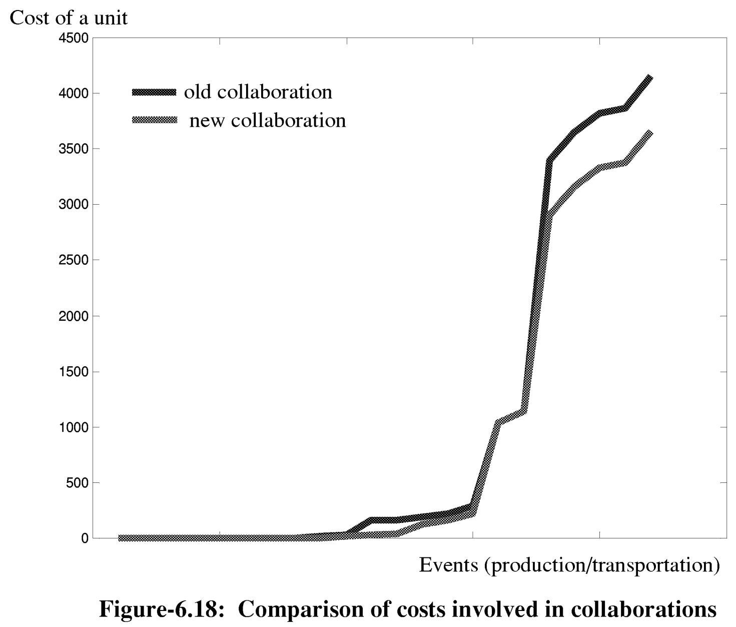


Figure-6.18: Comparison of costs involved in collaborations

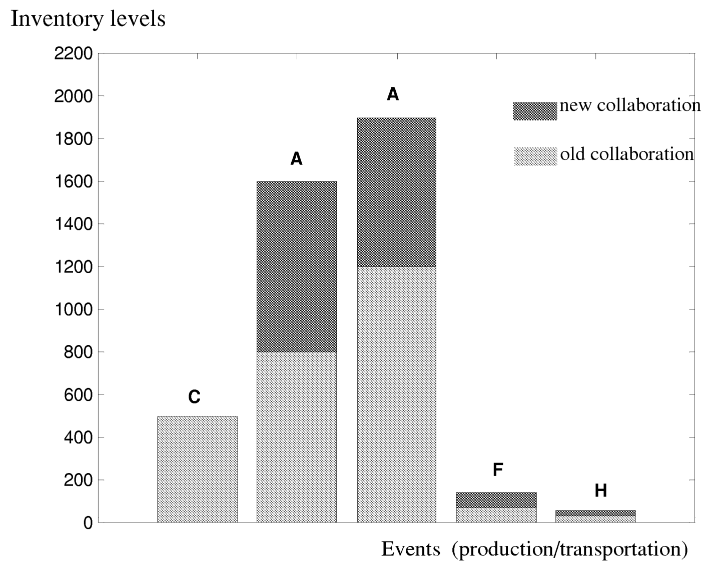


Figure-6.19: Comparison of inventory levels left in enterprises

6.10 SUMMARY

The last module for automating supplier selection procedures is the performance evaluation engine. The performance evaluation engine is a simple yet effective tool for performance evaluation of the supplier that is selected as the best supplier by the inference engine. The performance evaluation engine can be also used for performance measurement of the whole supply chain. The aim of performance evaluation engine is to put the new supplier in collaboration with the other existing collaborating enterprises for a specific project, and to carry simulations to see whether the supplier will perform satisfactorily in collaboration.

Realization of the performance evaluation engine is done in three stages. The first stage is the modeling stage where the manufacturing systems theory approach is used to establish a higher-level model (also called the connected system) of the real-life agile virtual enterprise. At the modeling stage, the tool AgileSIM is used to establish the connected system (AgileSIM is developed for MATLAB environment). The second stage is the simulation stage, where the connected system is converted into the lower-level Petri net model. This conversion is done automatically by AgileSIM. The tool PenSIM (Petri Net Simulator) can be used to carry-out simulations on the Petri net model. The third stage is the implementation stage at which the lower-level Petri net model is converted to C++ programming language code and compiled into a executable system - the performance evaluation engine. To model the lower-level we make use of timed colored Petri net.

7. CONCLUSION

Automating supplier selection procedures is highly beneficial to small and medium-sized agile virtual enterprise, because they can seek potential suppliers cheaper, faster and all over the world; thus, automating supplier selection procedures enables the enterprise to maintain its key concepts for survival, namely agility and virtual management.

To identify the steps of the supplier selection procedures that can automated, there is a need for a modeling approach with which one can model the supplier selection procedures of an enterprise; this paper presents a modeling approach that dissect the supplier selection procedures broadly into three stages; the pre-selection stage: Management sets the strategic goals for procurement. The selection stage: The main selection procedures, starting with many potential suppliers and ending with a most preferred supplier. The selection stage is further divided into three sub-stages called bidder selection, partner selection and performance evaluation. And the post-selection stage: Establishing collaboration with the selected supplier. The selection stages in the modeling approach is further divided into steps, as shown in tabular form below.

stage	***sub-stage***	***steps***
Pre-selection stage	Strategic goal setting	Management sets the strategic goals for procurement; also defines criteria such as low cost, JIT delivery, high quality etc.
Selection stage	on-site selection (1st level selection)	Make the request for proposal (RFP)/ request for quotes (RFQ); Receive quotes from suppliers and select a pool of potential suppliers satisfying the basic requirements (such as cost, quality, etc.)
	Partner selection (2nd level selection)	Analyze the supplier quotation and selection best suppliers based on the numerical calculation results
	Performance evaluation (3rd level selection)	The selected supplier is placed in a collaborative environment for a specific project, and performance evaluation is done to see whether the supplier will perform well in collaboration.
Post-selection stage	Selection of the most preferred supplier	Continuous communication with the selected supplier on materials, product development & testing, costing, etc.

Table-7-1: A basic modeling approach for modeling supplier selection procedures

By the methodology presented in this dissertation, only the steps within the selection are considered for automation. Pre-selection and post-selection stages are not considered for automation.

After the modeling approach, a methodology for automating supplier selection procedures is presented in this dissertation. By this methodology, each sub-stage of the selection stage (bidder, partner, and performance evaluation) employ a module for automation; the modules used for automation at these sub-stages are the data collection system, the inference engine and the performance evaluation engine respectively figure-7.1).

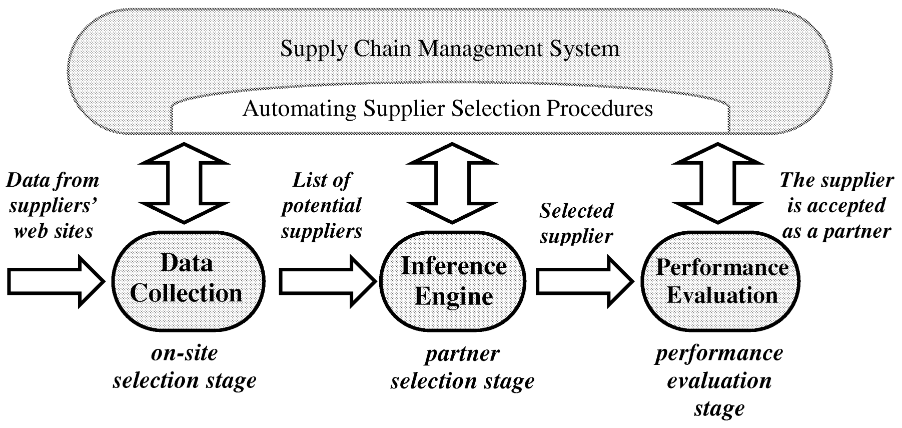

Supply Chain Management System
Automating Supplier Selection Procedures
Data from suppliers' web sites
Data Collection
List of potential suppliers
Inference Engine
Selected supplier
Performance Evaluation
The supplier is accepted as a partner
on-site selection stage
partner selection stage
performance evaluation stage

Figure-7.1: The three modules for automating supplier selection procedures

The data collection system described in this dissertation is the first of the three modules for automating supplier selection procedures; data collection system automates on-site selection sub-stage. The data collection system sends mobile agents to collects data (or quotes) from suppliers' web sites; to enable this, data provided by suppliers on their web sites must be structured-information using XML conforming to a publicly available uniform grammar. After reading the suppliers quotes, the mobile agents then accepts the supplier as a potential supplier and bring backs the data to the main assembler for further scrutiny, only if the supplier data satisfies the critical conditions sets by the assembler with broad margins. Because of mobile agents selecting suppliers on their web sites, this stage is called the 'on-site' selection stage. In-addition to collecting data from potential suppliers is for the formation phase of an agile virtual enterprise, the data collection system can be also extended to play the information infrastructure role during the operation phase too; this is also explained in this dissertation.

The inference engine is the second module for automation; the inference engine is for selecting the best supplier from a list of potential suppliers. Selection of the best supplier from the pool of potential suppliers is done in the sub-stage called the partner selection stage. The inference engine is realized with array-based logic. Array-based logic is chosen to realize the inference engine because, array-based logic offers fast computation, compact code, and complete solution. Fuzzy logic is another technology that can be used for realization of the inference engine, but fuzzy logic does not guarantee complete solutions.

This dissertation describes addition of the natural language processing capability to array-based logic when we wrote array-based logic in MATLAB system; the new toolbox of logic functions is called the structured array-based logic. With structured array-based logic, not only objective factors but subjective factors too (factors that can not be easily quantified) can also be included for evaluating the best supplier out of the pool of suppliers. However, for brevity, only the critical factors like price, delivery and quality are considered for selection by inference engine.

The last module for automating supplier selection procedures is the performance evaluation engine. The performance evaluation engine is a simple yet effective tool for performance evaluation of the supplier that is selected as the best supplier by the inference engine. The performance evaluation engine can be also used for performance measurement of the whole supply chain. The aim of performance evaluation engine is to put the new supplier in collaboration with the other existing collaborating enterprises for a specific project, and to carry simulations to see whether the supplier will perform satisfactorily in collaboration.

Realization of the performance evaluation engine is done in three stages. The first stage is the modeling stage where the manufacturing systems theory approach is used to establish a higher-level model (also called the connected system) of the real-life agile virtual enterprise. At the modeling stage, the tool AgileSIM is used to establish the connected system (AgileSIM is developed for MATLAB environment). The second stage is the simulation stage, where the connected system is converted into the lower-level Petri net model. This conversion is done automatically by AgileSIM. The tool PenSIM (Petri Net Simulator) can be used to carry-out simulations on the Petri net model. The third stage is the implementation stage at which the lower-level Petri net model is converted to C++ programming language code and compiled into a executable system - the performance evaluation engine. To model the lower-level we make use of timed colored Petri net.

Further work

In this work, the selection of collaborating partner is limited to suppliers; this is because, as far as SMEs are concerned, the supplier selection procedures for seeking optimum suppliers around the globe is much more important than seeking distributors overseas, as SMEs mainly satisfy domestic market. If further work is done on the methodology (chapter-3) for partner selection to include distributors too, then the modules for automation described in this dissertation can be extended to automate distributor selection too.

This work deals with automating the steps in selection stage only. Further work should be done on automating the other stages (pre- and post- selection stages) too, so that a fully automated supplier selection system results.

The other important work that can be done as a continuation of this dissertation, is a study about integrating the automated supplier selection system with the rest of the ERP (like BAAN etc.) system.

PUBLICATIONS

1. R. Davidrajuh and Z. Deng: "An Autonomous Data Collection System for Virtual Manufacturing System". *International Journal of Agile Management Systems*, Vol.2, No.1, May 2000, ISSN 1465-4652.
2. R. Davidrajuh and Z. Deng: "Identifying Potential Suppliers for Formation of Virtual Manufacturing System". Proceedings of the International Conference on Information Technology for Business Management (ITBM'2000), ISBN 3-901882-05-7 Phei Publication, Beijing-China, August 2000.
3. Z. Deng, R. Davidrajuh, B. Bang, and A. Lakså: "Multi-agent based modeling for inter-enterprise integration". Proceedings of the International Enterprise Modeling Conference (IEMC'99), in Verdal - Norway, June 1999.
4. R. Davidrajuh, Z. Deng and K. Wang: "Realizing Integrated Intelligent Manufacturing System with Distributed Intelligent Agents". Proceedings of the 15th International Conference on Computer-Aided Production Engineering (CAPE'99), ISBN 0-9535558-0-1, April 1999, Durham-UK.
5. R. Davidrajuh and Z. Deng: "Enabling Technologies for Information Flow in Computer Integrated Manufacturing Systems". Proceedings of the 18th AIMTDR Conference; Narosa Publishers, ISBN 81-7319-275-8, December 1998.

----

6. R. Davidrajuh: "A Petri Net Approach for Performance Measurement of Supply Chain in Agile Virtual Enterprises". *Submitted for publication* in the journal- "MIS Review". (October 2000).
7. R. Davidrajuh: "Hybrid Modeling Approach for Supplier Selection in E-Commerce". *Submitted for publication* in the Journal of Advanced Systems Studies. (August 2000).
8. R. Davidrajuh and Ø. Bjørke: "A Logic Toolbox for Manufacturing Systems Applications". *Submitted for publication* in the International Journal of Agile Management. (June 2000).

The publications are available on the web site: http://www.hin.no/~rd/phd.html

REFERENCES

Adcock, T. A. “What is Fuzzy Logic: An Overview of the Latest Control Methodology”, Texas Instruments, SPRA028, 1993

AgileSIM, “A Tool for Modeling and Simulation of Performance Measurement of Supply Chain Agile Virtual Enterprise”, http://www.hin.no/~rd/Projects/AgileSIM

Aglets (1999): www.trl.ibm.co.jp/aglets

Bee-gent (1999): www2.toshiba.co.jp/beegent

Bjørke, Ø., *Manufacturing Systems Theory*, TAPIR, Trondheim-Norway, 1995. ISBN 82-519-1413-2

Bjørke, Ø. *Modelling of Interdisciplinary Systems Design and Application.* To be published by CIRP publishers, 2000.

Bonney, M. C. “Are push and pull systems really so different? ”, *Int. J. Production Economics, vol. 59, pp. 53-64, 1999*

Bosak, J., and Bray, T. “XML and the Second-Generation WEB”, Scientific American, May 1999.

Boyd, M. “XML and Java Partner for Portable Internet Applications”, Internet Application Developer, Java Report, Vol. 4, No. 11, November 1999

Brown, P. and Gibson, D. “Systematic facility site selection monograph”, Dept. of Industrial Engineering, Montana State University, Bozeman, Montana-USA, 1980.

Bundesamt fur Statistik (1994) (Federal Bureau of Statistics), *Business and Employees on 25.05.1987* (In German)

Carter, J. R. and Narashimhan, R. “Purchasing in the international marketplace: implications for operation”, *Journal of Purchasing and Material Management*, Vol. 26, No. 3, 1990

Cassandras, C. G. and Lafortune, S., *Introduction to Discrete Event Systems*, Kluwer Academic Press, 1999. ISBN 0-7923-8609-4

Chick S. E. and Olsen, T. L. “A descriptive multi-attribute model for re-configurable machining system selection”, *Int. J. Agile Management Systems*, Vol.2, No.1, 2000.

Christopher, M., “Logistics and Competitive Strategy”, in: Cooper, J. (ed.), Strategy Planning in Logistics and Transportation, The Cranfield Management Research Series, 1993

Christopher, M., *Logistics and Supply Chain Management; Strategies for reducing Costs and improving Services*, 2nd Ed, Pitman Publishing, 1998

Concordia (1998): www.concordia.mea.com

Croom, S. *et al,* "Supply chain management: an analytical framework for critical literature review", *European Journal of Purchasing and Supply Management*, 6, pp. 67-83, 2000.

Dale, B. *et al, Supply Chain Management and Development* Prentice Hall, 1994

Davidrajuh, R. and Deng, Z: "Enabling Technologies for Information Flow in Computer Integrated Manufacturing Systems". Proceedings of the 18th AIMTDR Conference; Narosa Publishers, ISBN 81-7319-275-8, December 1998.

Davidrajuh, R., Deng, Z., and Wang, K: "Realizing Integrated Intelligent Manufacturing System with Distributed Intelligent Agents". Proceedings of the 15th International Conference on Computer-Aided Production Engineering (CAPE'99), ISBN 0-9535558-0-1, April 1999, Durham-UK.

Davidrajuh, R. and Deng, Z: "Identifying Potential Suppliers for Formation of Virtual Manufacturing System". Proceedings of the International Conference on Information Technology for Business Management (ITBM'2000), ISBN 3-901882-05-7 Phei Publication, Beijing-China, August 2000.

Davidrajuh, R., and Deng, Z. "An Autonomous Data Collection System for Virtual Manufacturing System", *the International Journal of Agile Management Systems*, Vol. 2, No.1, 2000, ISSN 1465-4652.

Deadman, R. "XML as a Distributed Application Protocol", Java Report, Vol. 4, No. 10, October 1999

Deng, Z. "A Model of Methodology Need for AQAL Production System", Proceedings of the International Conference on Integrated and Sustainable Industrial Production, Lisbon, Portugal, 1994.

Deng, Z., Bang, B., Laksa, A., and Nadarajah, S. "A Model of enterprise integration and collaboration tools and communication infrastructure for inter-enterprise collaboration", Globallization of Manufacturing in the Digital Communications Era of the 21st Century, Kluwer Academic Publishers, ISDN 0-412-83540-1, 1998

Deng, Z., Davidrajuh, R., Bang, B., and Laksâ, A.: "Multi-agent based modeling for inter-enterprise integration". Proceedings of the International Enterprise Modeling Conference (IEMC'99), in Verdal - Norway, June 1999.

Dickson, G. W. "An analysis of vendor selection systems and decisions", *Journal of Purchasing*, Vol.3, No.1, 1966

Dynamic Instruments Inc, http://www.dynamicinst.com

Enator (1998): Virtual Enterprising, http://195.100.12.162

Engelhardt, A. "A fresh look at integrated business applications - neither package nor custom-made! ", 8th Australasian Conference on Information Systems, 1997

Farhoodi, F. and Fingar P. “Developing Enterprise Systems with Intelligent Agent”, Distributed Object Computing, Nov. 1997

Fingar, P., Kumar, H., and Sharma, T. *Enterprise E-Commerce*, Meghan-Kiffer Press, Florida, ISBN 0-929652-11-8, 2000

Flanagan, D. *Java in a Nutshell*, 2ed, O'Reilly, 1997, ISBN 1-56592-262-X

Franklin, S. and Graesser, A. “Is it an Agent, or just a Program? ”, Proceedings of the 3rd Workshop on Agent Theories, Architecture, and Languages. Springer-Verlag, 1996

Franksen, O. I. “Group Representations of Finite Polyvalent Logic – a Case Study Using APL Notation”. In Niemi, A. (ed.): *A Link between Science and Applications of Automatic Control*, Pergamon Press, Oxford and New York, 1979.

Fuggetta A., Picco G. P., and Vigna G. “Understanding Code Mobility”**,** IEEE Transactions on Software Engineering, vol. 24, no. 5., 1998

Geannari, J. H. *Reuse, CORBA, and Knowledge-Based systems*, SMI, Stanford University, 1998.

Gensym. “Gensym’s ReThink and e-SCOR for Dynamic Business Process Modeling and Supply Chain Management”, White Paper by Gensym Corporation, http://www.gensym.com/, 2000

Ghingold, M. and Wilson, D. T., “Buying center research and business marketing practice: meeting the challenge of dynamic marketing”, *J. Business & Industrial Marketing*, Vol. 13, No.2, 1998

Goldman, S., Nagil, R. and Presis, K., *Agile Competitors and Virtual Organizations*, Van Nostrand Reinhold, New York, 1993

Goranson, H. T. *The Agile Virtual Enterprise: Cases, Metrics, Tools*, Quorum Books, 1999

Graham, P. G. “Small business participation in the global economy”, *European Journal of Marketing*, Vol.33, No.1/2, pp.88-102, MCB University Press, 2000

Gunasekaran, A., Marri, H. B., and Lee, B. “Design and implementation of computer integrated manufacturing in small and medium-sized enterprises: A case study”, *International Journal of Advanced Manufacturing Technology*, 16, 2000, Springer-Verlag.

Handford, R.B. and Nichols, E.L. *Introduction to Supply Chain Management*, Prentice-Hall, 1999

Harland, C. “ Supply chain management relationship, chains and networks”
British Journal of Management, 7 (Special Issue) pp. 63-80, 1996

Harhalakis, G., Proth, J. M., Savi, V. M. and Xie, X. “A stepwise specification of a manufacturing system using Petri nets”, Proceedings of the 1991 IEEE International Conference on Systems, Man and Cybernetics, Charlottesville, Virginia, October 1991

Huck, G., Fankhauser, P., Aberer, K., and Neuhold, E. “Jedi: Extracting and Synthesizing Information from the Web”, the 3rd IFCIS International Conference on Cooperative Information Systems (CoopIS'98), New York, August 1998

Humphreys, P., Mak, K. L. and Yeung, C. M. “A just-in-time evaluation strategy for international procurement”, *Supply Chain Management: An International Journal*, Vol.3, No.4, 1998.

Hunt, I., Carelli, R., Pellicar, J., Gonzalez, I. M. Chacon, V. H. “Modeling to Support a Food Industry in the Extended Enterprise”, Proceedings of the CAPE’97, Detroit - USA, November 1997.

Hussein, B. *On modeling mechatronic systems: a geometrical approach.* Department of Production and Quality Engineering, Norwegian University of Science and Technology, Report NTNU 97007, 1997

Houshyar, A. and Lyth, D. “A systematic supplier selection procedure”, *Computers in Industrial Engineering*, Vol.23, No.1-4, 1992.

InstantDB (1999): www.instantdb.co .uk
Jannarone, R.J. *Concurrent Learning and Information Processing: A Neuro-computing System that Learns during Monitoring, Forecasting, and Control.* New York: Chapman & Hall, 1997

Jannarone, R.J. “Locally dependent models: conjunctive item response theory”. In W.J. van der Linden & R.K. Hambleton, III (eds.) *Handbook of Modern Item Response Theory*, New York: Springer-Verlag, 1997

Jeyakumar, P. *A Design Approach to integrate intelligent systems in a manufacturing environment*, Ph.D. Dissertation, Norwegian University of Science and Technology (NTNU), Norway, 1995.

Kaplan, R. S. and Norton, D. P. “The Balanced scorecard - Measures that drive performance”, Harvard Business Review, January-February, 1992

Kjenstad, D. *Coordinated Supply Chain Scheduling*, Ph.D. dissertation, Norwegian University of Science and Technology (NTNU), Trondheim-Norway. Doc. id. 1998:24, 1998

Kojima, T., Sekiguchi, H., Nakahara, S., and Ohtani, S. “An Expert System of Machining Operation Planning in Internet Environment”, the 15th International Conference on Computer-Aided Production Engineering (CAPE'99), Durham, April 1999

Kusiak, A. “Knowledge-Based Systems”, Nordic-Baltic Summer School on Applications of AI to Production Engineering, Ed. K. Wang, Kaunas University of Technology Press, 1997

Kvaerner Oil and Gas (2000): http://www.kvaerner.no/oilgas/

Lapide, L. “What about measuring supply chain performance”, White Paper by AMR Research, http://www.amrresearch.com/, 2000

Laufmann, S. C. “Agent Software for Near-Term Success in Distributed Applications”, *AGENT TECHNOLOGY - Foundations, Applications, and Markets*, Springer-Verlag, 1998, ISBN 3-540-63591-2.

Leventhal, M., Lewis, D., and Fuchs, M. *Designing XML Internet Applications*, Prentice-Hall PTR, 1998, ISBN-0-13-616822-1

Levy, A. L. “The information Manifold approach to data integration”, IEEE Intelligent Systems and their applications, September-October 1998, Vol. 13, No. 5

Li, H. “XML and Industrial standards for Electronic Commerce”, *Knowledge and Information Systems, vol. 2, pp. 487-497, 2000*

Li, Y., Huang, B., Liu, W., Wu, C. and Gou, H., “Multi-agent system for partner selection of virtual enterprise”, Proceedings of the Int. Conference on Information Technology for Business Management, Beijing, China, August 2000

Liu, J., Ding, F. and Lall, V. “Using data envelopment analysis to compare suppliers for supplier selection and performance improvement”, *Supply Chain Management: An International Journal*, Vol.5, No.3, 2000.

Marquardt, E. P. “Aligning Strategy and Performance with the Balanced Scorecard: An Interview With David P. Norton”, *ACA Journal, the quarterly publication of the American Compensation Association*, Volume 6 Number 3, Autumn 1997

Maruyama, H., Uramoto, N., and Tamura, K. *XML and Java: Developing Web Applications*, Addison-Wesley, 1999. ISBN 0-201-48543-5

MATLAB (2000): Introduction. http://www.mathworks.com/products/matlab/

Meridian Marketing Group, *Fuzzy Logic Newsletter*, Vol.2, No.1, 1997.

Min, H. and Galle, W. P. “International purchasing strategies of multinational US firms”, *International Journal of Purchasing and Material Management*, Vol. 27, No.3, 1991

Morgenthal, J. P. “XML solutions: Comparing ebXML and UDDI”, XML Journal, November 2000, Vol. 1, Issue. 6

Motwani, J., Youssef, M., Kathawala, Y., and Futch, E. “Supplier selection in developing countries: a model development” in *Integrated Manufacturing Systems*, Vol.10, No.3., 1999

Møller, G. L. *On the Technology of Array-Based Logic*, Ph.D. Dissertation, Technical University of Denmark, 1995

Nagel, R. and Dove, R. *21st century manufacturing enterprise strategy*, Iacocca Institute, Leigh University, US. 1991.

Nwana, H. S. and Ndumu, D. T. “A Brief Introduction to Software Agent Technology”, *AGENT TECHNOLOGY - Foundations, Applications, and Markets*, Springer-Verlag, 1998, ISBN 3-540-63591-2.

Orfali, R. Harkey, D. and Edwards, J. “CORBA, Java, and the Object Web”, Byte, October 1997

Barbuceanu, M. and Fox, M. S. "Coordinating Multiple Agents in the Supply Chain". WET ICE'96.

Pries, K. "The Emergence of the Interprise", in Schonsleben, P. and Buchel, A. (eds.), *Organizing the Extended Enterprise*, Kluwer Academic Publishers, 1998

Rembold, U., Nnaji, B. O., and Storr, A. *Computer Integrated Manufacturing and Engineering*, Addison-Wesley, 1993, ISBN 0-201-56541-2

RmiJdbc (1999): http://www.objectweb.org/RmiJdbc/RmiJdbcHomePage.htm

SABL (2000): An Introduction to SABL. http://www.hin.no/~rd/Projects/SABL

Savi, V. M. and Xie, X. "Liveness and Boundedness Analysis for Petri nets with Event Graph Modules", *Petri nets*, *Lecture Notes in Computer Science series*, Springer-Verlag, 1992

SCC (Supply-Chain Council, Inc): www.supply-chain.org

Schmidt, D. C., Levine, D. and Mungee, S. "The Design of the TAO Real-Time Object Request Broker", *Computer Communications, Special Issue on Building Quality of Service into Distributed Systems*, Elsevier Science, Volume 21, No. 4, April, 1998.

Shank, J. and Govindarajan, V. *Strategic Cost Management: The new tool for Competitive Advantage*, Free Press, 1993

Shindler, G., and Lamprecht, J. "E&P companies, suppliers move to international standards", *Oil & Gas Journal*, Vol. 89, No. 18, 1991

Silva, M., Teruel, E., Valette, R., and Pingaud, H. "Petri nets and Production Systems", *Lectures on Petri Nets II: Applications*, *Lecture Notes in Computer Science 1492*, Springer-Verlag, 1998

Sihn, W., Deutsch, O. and Haege, M. "Supply chain management for small and medium-sized enterprises", Managing Innovative Manufacturing Conference (MIM2000), Aston Business School, U.K.

Slack et al, *Operations Management* 2nd Ed. Pitman Publishing, 1998

Sommers, B. "Agents: Not just for Bond anymore". Java World, April 1997

Stevens, G. "Integrating the supply chain", *International Journal of Purchasing and Materials Management*, 19 (8) pp. 3-8, 1989

Storey, D. *Understanding the Small Business sector*, Routledge Publishers, 1994

Sundsted, T. "An introduction to agents". Java World, June 1998

Sundsted, T. "Agents on the move". Java World, July 1998

Sundsted, T. "Agents talking to agents", Java World, September 1998.

Thompson, K. N. “Vendor profile analysis”, *Journal of Purchasing and Material Management*, Vol. 26, No.1, 1990

Towers, N. “Execution of short term production planning with virtuous manufacturing: Towards a paradigm for small and medium sized enterprises operating in a supply chain”, Managing Innovative Manufacturing Conference (MIM2000), Aston Business School, U.K.

Tsoukalas, L. H. and Uhrig, R. E. *Fuzzy and Neural Approaches in Engineering*, John Wiley and Sons, 1997

US Small Business Administration (1992): *The state of small business, a report of the president of the US Congress.*

Venners, B. “The architecture of Aglets”. Java World, April 1997

Vogel, A. and Duddy, K. *JAVA Programming with CORBA*, 2 ed., John Wiley, 1998

Object Management Group http://www.omg.org

Voyager (1999): http://www.objectspace.com

Walsh, T., Paciorek, N., and Wong, D. “Security and Reliability in Concordia”, Proceedings of the 31st Annual Hawaii International Conference on System Sciences (HICSS31), Hawaii, January 1998.

Wang, K. “A New Modeling and Analyzing Approach to Material Flow and Productivity”, International IFIP Conference on Computer Applications in Production and Engineering, Beijing, China, 1995

Weber, C. A., Current, J. R., and Benton, W. C. "Vendor selection criteria and methods”, *European Journal of Operations Research*, North-Holland, Vol.50, 1991

Wolfe, V. F. “Real-Time CORBA”, Proceedings of the Third IEEE Real-Time Technology and Applications Symposium, Montreal, Canada, 1997.

Womak, J. P., Jones, D. T., and Roos, D. *The machine that changed the world: the story of lean production.*, 1st HarperPerennial, 1991, ISBN 0-06-097417-6

Wong, D., Paciorek, N., Walsh, T. “Concordia: An Infrastructure for Collaborating Mobile Agents”, First International Workshop on Mobile Agents (MA'97), April 1997, Berlin, Germany.

Woodside, A. G., Liukko, T. and Vuori, R. “Organizational buying of capital equipment involving persons across several authority levels”, *J. Business & Industrial Marketing*, Vol. 14, No.1, 1999

XML4J (1999): www.ibm.com/developer/xml/

Yager, R. R. and Zadeh, L. *An Introduction to Fuzzy Logic Applications in Intelligent Systems*, Kluwer Academic Publishers, 1991

Yam, R., and Lo, W., and Tang, P. Y. “Enhancement of global competitiveness for Hong Kong/China manufacturing industries through i-agile virtual enterprising”**,** Managing Innovative Manufacturing Conference (MIM2000), Aston Business School, U.K

Yellow Pages Australia, *Small Business Index: A special report on small business growth aspirations and the role of exports*, February 1995, Melbourne

Yousef, M. A. “A decision support system for evaluating and selecting suppliers in an advanced manufacturing technology environment”, National Meeting of the American Institute for Decision Sciences, San Francisco, November 1992.

Yousef, M. A., Zairi, M. and Mohanty, B. “Supplier selection in an advanced manufacturing technology environment: an optimization model”, *Int. J. Benchmarking for Quality Management and Technology*, Vol.3, No.4, 1996.

APPENDIX-1: Software packages

During this Ph.D. research, I developed three software packages; the first one is the structured array-based logic, a toolbox of logic functions. The second package is AgileSIM/PenSIM, which is used for performance evaluation of a potential supplier in collaboration. The third package is a simple non-graphic simulator for general Petri nets, GPenSIM.

A1. SABL

Acronym SABL stands for Structured Array-Based Logic. SABL is developed for MATLAB simulation environment. SABL is a collection of logic functions for propositional logic and array-based logic. With MATLAB, I wrote SABL in such a manner that computing with words (like fuzzy logic) is possible; the SABL functions and a user manual is available at:

http://www.hin.no/~rd/Projects/SABL

A2. AgileSIM

The simulation software AgileSIM is used for modeling a supply chain of an agile virtual enterprise; AgileSIM is also used for performance evaluation of a new supplier when the supplier is placed in collaboration with the rest of the collaborating enterprises of an agile virtual enterprise. Use of AgileSIM is discussed in chapter-6, "Performance Evaluation". The AgileSIM package is available at:

http://www.hin.no/~rd/Projects/AgileSIM

The toolbox AgileSIM also has an integrated tool called PenSIM which is a special purpose Petri Net simulator based on timed colored Petri net. PenSIM is used to convert a Petri net into C++ code so that the a Petri net model can be converted into a executable program; see chapter 6 for further explanation.

A3. GPenSIM

GPenSIM is a simple general purpose Petri Net simulator. GPenSIM is not a graphic simulator, the output of simulations are ASCII files or matrix. GPenSIM is available at:

http://www.hin.no/~rd/Projects/GPenSIM

ACRONYM

ACA	Accounting Agent
ACID	Atomic, Consistent, Isolated, and Durable
API	Application Programming Interface
CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
CAP	Computer Aided Planning
CAQ	Computer Aided Quality Control
CIM	Computer Integrated Manufacturing
CORBA	Common Object Request Broker Architecture
CPM	Critical Performance Measure
CR	Component Ratio
DCA	Distributor Coordinator Agent
DOM	Document Object Model
DTD	Data Type Definition (XML)
GUI	Graphical User Interface
HTML	Hyper Text Markup Language
IDL	Interface Definition Language (CORBA)
IP	Internet Protocol
JDBC	Java DataBase Connectivity
JIT	Just-In-Time
JPDA	Just-in-time Procurement and Distribution Agent
JVM	Java Virtual Machine
KPI	Key Performance Indicators
MA	Mobile Agent
MCA	Main assembler Coordinator Agent
MRO	Maintenance, Repair and Operating
MRP	Manufacturing Resource Planning
ORB	Object Request Broker
OTS	Object Transaction Service (CORBA)
PDA	Product Design Agent
PMA	Product Manufacture Agent
PP&C	Production Planning and Control
SAX	Simple API for XML
SCA	Supplier Coordinator Agent
SCM	Supply Chain Management
SCOR	Supply-Chain Operations Reference
SME	Small and Medium-sized Enterprise
SSLv3	Secure Socket Layer - version 3
URL	Uniform Resource Locator
VCA	Virtual enterprise Coordinator Agent
WIP	Work-In-Progress
WWW	World Wide Web
XML	eXtended Markup Language

LIST OF FIGURES

LIST OF TABLES