Dynamic Enterprise Modelling
A Methodology for Animating Dynamic Social Networks

by

Panayiotis Periorellis

Dynamic Enterprise Modelling

A Methodology for Animating Dynamic Social Networks

by

Panayiotis Periorellis

Presented to the Department of Computing Software Reliability of

The University of Newcastle

In partial Fulfilment of the Requirements

for the Degree of

Doctor of Philosophy

University of Newcastle

May 2000

Abstract

Since the introduction of the Internet and the realisation of its potential companies have either transformed their operation or are in the process of doing so. It has been observed, that developments in I.T., telecommunications and the Internet have boosted the number of enterprises engaging into e-commerce, e-business and virtual enterprising. These trends are accompanied by re-shaping, transformation and changes in an enterprise's boundaries. The thesis gives an account of the research into the area of dynamic enterprise modelling and provides a modelling methodology that allows different roles and business models to be tested and evaluated without the risk associated with committing to a change.

Acknowledgments

The author would like to thank the following:

Prof. John E. Dobson for supervising this Ph.D. research and providing invaluable support and guidance.

Prof. Peter Smith for his support during the first year of the research.

Dr. Albert Bokma for his supervision during the initial stages.

N.D.C. Northern Development Company for proving the requirements of the prototype system.

Ms. Kate M. Jones for helping me to write chapter 5 and proof reading.

TABLE OF CONTENTS

CHAPTER 1 General Introduction

CHAPTER 2 The Need for Enterprise Modelling

CHAPTER 3 Literature Review

CHAPTER 4 Dynamic Method

CHAPTER 5 Knowledge Representation and Inference

CHAPTER 6 Implementation

CHAPTER 7 Summary and Conclusions

CHAPTER 8 References and Bibliography

FIGURES-List of Figures

TABLES-List of tables

Chapter 1 General Introduction

1.0 Introduction

Networks of communicative agents are inter-connected entities grouped together under a series of individual and network objectives. The term *'networks'* indicates that there is a set of connections between these agents. The connections represent the relationships that exist between them, as well as the degree of dependency. When these agents take the form of organisational structures, we know that the relationships between them affect the formulation of strategies, and determine the way they operate at a strategic, operational and tactical level.

On a larger scale the same can be said of individual organisations themselves. All organisations operate towards a set of objectives. These objectives affect, and are affected by other organisations. We refer to these organisations as enterprises and the connections between them as dependencies. Although there have been considerable efforts made in modelling enterprise networks in general, there has not been enough effort dedicated to the animation of large scale networks of agents.

The author has identified that the area of animating networks of interconnected agents as an area worthy of attention and carried out a research in the area modelling and animating those agents. This thesis presents the findings of this research focusing on the methodology for modelling and animating networks of agents. The methodology has also been used to develop a prototype that is described in detail in Chapter 5. The overall aim of this research has been to carry out an investigation into the area of enterprise modelling and to produce a methodology for modelling and animating enterprises. Finally it provides the means of assessing the impact of changes on a network caused by animation or movement. In the following sections the chapter provides an overview of enterprise models with particular emphasis to dependency modelling as it is mostly related with thesis, and explains what their aims are and what they target.

1.1 An Overview of enterprise dependency modelling

Enterprise dependency modelling deals with the relations between enterprises and how they affect each other's operations. The impact of these relations could affect decisions such as *'What supplier should we choose ?'*, up to strategic level decisions such as '*Where should the new plant be built ?*' . These decisions are directly related to the services of an enterprise which in the case of a product company are, generally speaking:

- procurement,
- production (although it does not apply to service companies),
- marketing and
- distribution or service delivery.

Figure 1.1 depicts a typical a group of enterprises in a supply chain.

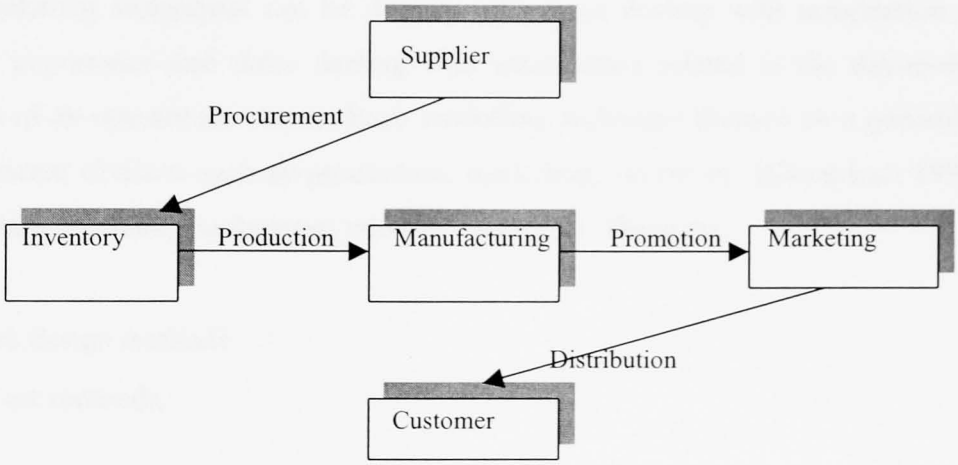

Fig 1.1 A Typical Supply Chain

The service departments within an enterprises follow their own goals and objectives. These objectives are often in conflict. Managing an enterprise network requires participation in a series of decision-making processes that are related to the network. Most of the decision making process is related to location, production, inventory and transportation. For each one of these categories there are data to be taken into account.

Assuming we perform some decision making regarding location, then we need to know about appropriate geographic locations and how this affects the overall performance of the organisation. We also need to know about inventory locations as well as sources of material. The decision would then be based on an assessment of all of the alternatives. On the other hand, for a production decision we would need to consider what products to produce and at which plants, as well as which suppliers will be allocated to each plant. Similar processes have to be performed for the other categories mentioned above. All of these have a direct impact on the organisation's operations as well as on revenue, costs, customers and markets.

1.2 Generic Modelling Techniques

Modelling enterprise dependencies is the process of identifying those factors that directly affect an enterprise's operations and the degree of change they cause. There have been several attempts in the past to model enterprise networks. Each one of these techniques attempts to assist in the decision making process with regard to a level of operation (tactical, operational, strategic) within an enterprise. Generally speaking, these modelling techniques can be divided into those dealing with information of strategic importance and those dealing with information related to the day-to-day activities of an operational nature. Each modelling technique focuses on a particular organisational division such as production, marketing, inventory. [Ganenham 1995] divides these modelling techniques into two categories. These are:

- Network design methods
- Rough cut methods,

1.2.1 Network Design Methods

Network design techniques tend to focus on modelling methodologies that relate to location of production, stocking facilities, sourcing facilities as well as the relationships and dependencies between them. There have been three approaches to the development of such models.

The first method, by [Cohen 1985] called *PLANETS*, attempted to provide a framework for modelling production and distribution. The model focused on what

products to produce, and which plants and suppliers should be used. General Motors implemented a subset of this approach [Ganenham 1995]. [Breitman 1987] focused on distribution modelling. In other words, he concentrated on modelling activities such as product flow, material produced and flow of materials from the suppliers through to the final customers. The third approach, was by [Cohen 1989], who presented a model for resource deployment in a global production and distribution network. The model catered for global raw material flow within a supply chain network, taking into account variables such as procurement, distribution and transportation.

A further development took place in the 1990's by Arntzen, Brown, Harrison and [Artzen 1997]. The model focused on two elements of strategic importance: cost and time. Bearing in mind the objective to minimise cost and time they provided a model of the major activities of an organisation such as purchasing or manufacturing. The model has received extra attention because of the consideration it has given to time and cost and also because it provides a global view.

1.2.2 Rough Cut Methods

These methods tend to focus on activities of a tactical or an operational nature, and usually take an inventory perspective. They deal with inventory control functions and they consider several levels of inventory together. IBM developed a model to manage its spare parts inventory. The model was called '*Optimizer*' and IBM claimed it was the most complex up to that date. However this model as well as further developments didn't take into account the production process. This is where they actually drew the line. It was capable of managing finished goods stock, regarding distribution and sales, but it was limited as a decision support model, as it lacked a wider business view. The difference with these methods and the approach presented in this thesis is that we do not distinguish between functions. In generic terms we're concerned with the nature of relationships between agents, which can take the form of systems, system's components, or enterprises. Our aim contrary to the methods mentioned above is not to improve the services of a network of agents but to allow exploitation of new states in terms of roles and responsibilities.

1.3 Overview of the thesis

After this brief introduction to the area of enterprise modelling it is now appropriate to outline the problem area, the research carried out, the findings and the developments made. Enterprises today after realising the potential of the Internet are going through a stage of transformation. Technologies such as electronic commerce or electronic businesses set out a new space of possibilities but they also demand changes in the roles and responsibilities of enterprises. Modelling in general is a way of exploring a new space and at the same time keeping the risk factor low. Dynamic modelling in particular would allow for different scenarios and roles to be tested and evaluated, without the risk associated with committing to a change.

The aim of the research is the development of a methodology for modelling and animating enterprise dependencies. Bearing in mind the extensive research that has been done on specific subjects such as supply chain networks as well as on enterprise modelling in particular [Sadeh 1996], I looked at an area that hadn't received much attention to date. One area that has being overlooked (perhaps due to the vast amounts of information that are associated with it), is the area of dynamic modelling.

This thesis describes the research carried out in the field of enterprise modelling. During the research I found several methods that used the term enterprise modelling for different purposes, as well as methodologies that referred to or flirted with the area of enterprise modelling. The findings of this research are presented in Chapter 3. These findings are accompanied by some degree of criticism as well as a degree of assessment of their relevance to the research carried out. In Chapter 4 I present a methodology for animating enterprise models in conjunction with a set of requirements; in terms of information that needs to be obtained in order to build such a model.

The development of a software prototype illustrated many of the ideas presented in Chapter 4. This system is described in Chapter 5. Chapter 6 discusses questions that arise from Chapter 4. Chapter 7 concludes the thesis.

Chapter 2 The Need for Enterprise Modeling

2.0 Introduction

Since the introduction of the Internet and the realisation of its potential companies have either transformed their operations or are in the process of doing so. Some companies use the net to connect with their customers while others offer trading facilities. It has been proved that the net has along with development in I.T. and telecommunications enabled companies to lower costs across their supply chains, amend their business processes or even change their role and responsibility on a supply chain. The above technologies have boosted the number of companies engaging in electronic commerce e-business and virtual enterprising. The changing roles and responsibilities imply different input/output functions that also have an impact across a supply chain. A recent survey carried out by the Economist [Economist 1999] showed that over 90% of businesses already have some form of presence on the web, while a large proportion is considering re-shaping, transformation, and change on their boundaries. It is not just the Internet however. Companies find it a lot easier to outsource or to develop short term relations with other businesses for mutual benefit. It becomes apparent after studying a particular market of industrial sector that companies are implementing networks of small businesses or individual all bound by the same corporate culture and communications. Once a Hollywood studio employed everyone from the leading actors to the lighting technicians. Nowadays studios assemble teams of self employed individual and independent businesses that are today's stars and technical support. Because the roles of firms is changing we need a model to assess the impact of changing roles, relations, formation of new businesses and roles.

The objective of this chapter is to define the term "enterprise model" along with all the related terms used throughout the thesis, determine what makes a successful model and why there is a need for a dynamic enterprise model. The author first attempts to define all the terms used in the thesis such as enterprise, process, dynamics etc. It has been found out during this research that there is a number of enterprise modeling definitions provided by different authors, that describe enterprise modeling in different contexts. In the following paragraphs some of these definitions are presented along with some critical

analysis as to what they imply or refer to. Finally the author presents his own definition of the term, and makes the case for a dynamic model.

2.1 Definitions of terms

In the following paragraph the author presents some definitions of the most common terms used throughout the thesis. The word enterprise in the context of this thesis, implies any organisation that is engaged in any type of activity i.e. manufacturing or simple information processing. Some examples of enterprises are banks, insurance companies, ticket reservation systems, production systems, factories, etc. In a general sense enterprise modelling is a way of precisely describing various aspects of an enterprise. *Precise* means describing the enterprise in a way which is clearly understandable to people. Actually, the goal of enterprise modelling is very broad. It covers all the aspects ranging from equipment and computer systems, to human resources, manufacturing, distribution, marketing, sales etc. There are however a number of definitions that describe enterprise modelling. In the following paragraphs of this chapter the author reviews some of the definitions and concludes a definition that describes the term within the context of this research. One can say that enterprise modelling is a new type of programming in which the executor of the "program" is not the computer, but rather the enterprise as a whole; it is made up of employees, as well as equipment (including computers) [Barzdins 1997].

One may ask why is enterprise modelling so important, and why is it that this problem has become so current lately. It has been concluded during this research that the main reason for enterprise modelling is that the structure of enterprises and the algorithms of functioning have become very complicated, (i.e the operations of a bank). At the same time, because of the introduction of computers in enterprise work, intuitive solutions and algorithms are no longer sufficient to make sure that the enterprise operates successfully. In a different level of detail one can also observe the complexity of computing systems as well as incompatibility issues. In conjunction with the above can be shown the need for process re-engineering in order to keep up technological updates. Gradually it becomes apparent that there is a need for a formalised procedure for developing schematics of an enterprise's processes.

So how is an enterprise process defined? According to a widely accepted definition [Hammer 1998] a business process is "a set of activities that produces a result which is valuable from the point of view of the customer or buyer". Business process re-engineering as it is discussed in the next chapter provides guidelines for improving business processes. The main question of BPR is not to improve the process but to conclude whether this process is needed (whether it produces something attractive for the customer). Less revolutionary definitions, which are also widely held, of what a business process is, e.g. business process is not simply a set of activities, but it also represents the sequence of these activities (process) which produce the expected result; Another point concerns the receiver of the result. This can be both an external agent (customer or buyer) and also another organisation unit within the same system. [Barzdins 1997] Business processes are often related with BPR under the supervision of an enterprise modelling methodology.

Enterprise models in general serve a particular purpose. Sometime they are used to demonstrate a business process while often they are used to re-engineer certain aspects of an enterprise's operations. Researchers and authors who work on the field of enterprise modelling define enterprise according to the use, view or perspective of the enterprise model. Before reviewing the various definitions the following paragraph describes the various perspectives enterprise models incorporate.

Descriptive enterprise models demonstrate the current operations and business functions of a system. For example the first level of IDEF:3 models would come under this category as it describes how the systems are currently working. Chapter 3 examines the IDEF methods in detail. Prescriptive models described how the world should be. These are the types of model we use for communicating a system's design. Consider for example the models we develop using SSADM. The series of models developed using this method are used for communicating the systems requirements between users and analysts. Constitutive models are used for explanatory purposes such as data flow diagrams that are used to develop models of data exchange between the various system's components. Optative models show what is desired. This is usually the case between users and systems analysts during the stage of capturing the requirements of the system. The users or group of users would normally draw a picture of how the system should work and what is desired (optative model) from the system and the analyst would then, based on the

user's view draw a prescriptive view of the system. We can conclude, based on the discussion so far, that enterprise models are based on a variety of perspectives. The perspective details the elements that the analyst of the model selects to incorporate in the model, in other words the nature of questions to which the model can provide answers. Based on these different perspectives of models the thesis classifies the enterprise modelling methodologies and tools which are described in Chapter 3, according to their ontological scope and dynamics.

As it is shown in chapter 3 the development of enterprise models is based on specification languages which form part of a methodology. These specification languages are used to service the design process that pre-dates programming. First and foremost, the languages are distinctly graphical in nature. System descriptions in a formal language are useful only if the descriptions are easily perceived, if they help to understand the system much better than does the natural-language description of the same system. The graphical nature of the description (currently in a two-dimensional space) is one of the best ways to make easier the perception of a system. It is not an accident that when people explain certain concepts, they often use graphic elements. The new generation of specification languages has turned these elements into official descriptive resources with precisely defined semantics. Obviously one of the major achievements in the area of specification languages over the last several years has been the Object Modelling Technique developed by [Rumbaugh 1991]. The basic idea of the OMT is very simple: systems must be described with simple (and as narrowly constructed as possible) sentences, and these sentences must be presented graphically. Moreover, each noun must be represented by a single node. Take a look at these sentences:

"A bank holds accounts". "A customer has one or more accounts in the bank". "A customer can commission a transaction". "A transaction concerns a specific account". "A transaction may be either a withdrawal or a pay-in". "The bank consists of the head office and several divisions". "The bank owns the central computer". "Divisions own their own PCs".

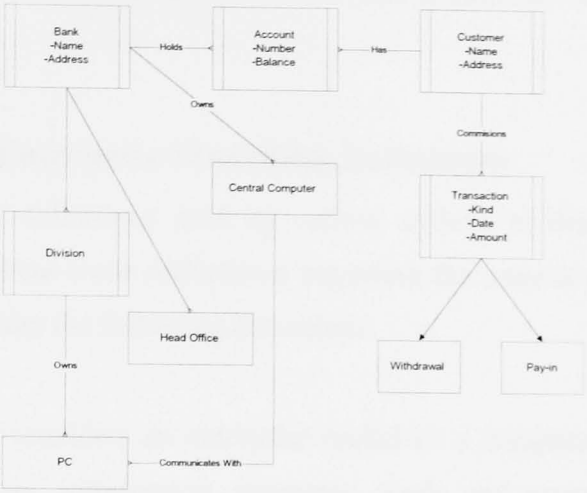

Figure 2.1 Example of OMT diagram

If we present a complex system in OMT form (which requires a bit of training), it becomes easily reviewed and understood. Even though we are not cognisant of the processes that are taking place in our brains as this happens, we can nevertheless conclude that to understand a system means to create a model of the system in our consciousness (our brain) which is similar to that which we see in an OMT image. If the description of the system is presented in natural language (i.e., in linear text), then it takes us a while to "translate" it into OMT form in our brain. One can say that the graphical languages serve to shorten this translation process quite considerably.

Of course, OMT can be used only to describe the static structure of an enterprise. What follows is the most essential part of the process: formalisation of the business processes in which the company is engaged. The term "business process" has been accepted recently to describe the chain of activities that an enterprise performs in order to reach its goals. These activities may include work done by people, as well as work done by computers and other technical equipment. Again we are faced with the question of the language in which these business processes are "programmed". It is clear that traditional programming languages are useless in this purpose, because in that case we must program not just the computer, but the entire enterprises, complete with the people who perform tasks at a much higher level. Technological achievements, the vast pace of globalisation and the

changing role of business agents calls for a more dynamic approach into the modelling of an enterprise's aspects. The author examines these developments in details later in the chapter. First however, we examine some of the definitions of the enterprise modelling term.

2.2 Review of Enterprise Modelling Definitions

There are several definitions used by various authors to describe the term enterprise modelling which bear some differences regarding the aspects of the enterprise they are referring to. Consider the following definitions.

[Gruninger 1996] considers an enterprise model as a computational presentation of the structural, processes, information, resources, goals and constraints of a business, the government activity, or the organisational system. It can be both definitional and descriptive, spanning what should be and what is. The role of an enterprise model is to achieve model driven enterprise design, analysis and operation.

[Liles 1996] believes that enterprise modelling is the body of knowledge, principles, and practises having to do more with the analysis, design, implementation and operation of an enterprise. In a continually changing and unpredictable competitive environment the enterprise modeller addresses a fundamental question: "how to design and improve all elements associated with the total enterprise through the use of engineering and analysis methods to more effectively achieve its goals and objectives".

[Whitman 1997] makes the following observations. A model is an abstract representation of reality. The modeller determines which aspects of the real system are of interest and are to be modelled. An enterprise is a complex system of cultural, processes and technology components. It is a system engineered to accomplish complex organisational goals. Therefore, an a enterprise model is defined as "a symbolic representation of the enterprise and the things that it deals with. It contains the representations of individual facts, objects, and the relationships that can occur within the enterprise."

[Vernadat 1994] concludes that enterprise modelling and analysis methods, tools and methodologies to support system design and to prepare system implementation according

to a system requirement are definitely required by the industry for the implementation of integrated systems. Therefore enterprise modelling attempts to achieve full system integration.

[Feldner 1999] Enterprise Models are software-based representations of an enterprise's products, markets, technology and support resources. These models enable CEOs, Directors and Managers to plan, assess and enhance the potential success of their business.

[Fox 1996] claims that an enterprise model is the language used to explicitly define an enterprise. It allows the systems analysts to explore alternative models in the design of the enterprise planning, organisation, structure and behaviour.

One can notice that there are clear differences between Gruninger's definition who refers to computational representations of enterprise operations and Liles who refers to enterprise modelling as a tool for achieving strategic objectives. Furthermore Fox considers an enterprise model as a tool for exploring alternative system design unlike Vernadat who claims enterprise modelling is a method for achieving systems integration. The distinctions exist because authors perceive enterprise modelling from different perspectives. This also proves the point made earlier in this chapter about different types of enterprise modelling and different model perspectives. The following paragraphs distinguish between the definitions supplied and discusses their views as to what enterprise modelling is and how it should be used.

2.3 Definition of Enterprise Modelling

The previous paragraphs showed that there are many commercial and non-commercial projects that come under the heading of enterprise modelling, although in fact they differ in their content and aims. During this research we identified 3 distinct types of enterprise models. Process oriented models concentrate in the modelling of activities for the purpose of performance measuring or business process reengineering. Resource oriented models although sometimes used in conjunction with process models target the input and output procedures a process or activity. Although this type of model can also be used for BPR purposes it can also facilitate the process of lean manufacturing were the aim is to find the optimum I/O combination by maximising output with minimum input. A resource

oriented prototype model has been developed for the purpose of this research and it is presented in chapter 6. Structure-oriented models emphasise on the roles and responsibilities of the participating agents. A good reason for developing such models would be to experiment with new structures or business models. It has been concluded that this type of modelling has received the least attention.

In a general sense the developments, methodologies and research projects defined by these definitions can be divided into two broad categories. The first category defines an enterprise model as a computational representation of an enterprise's activities, resources, costs, time scales, etc.[Fox 1996] whereas the second describes an enterprise model as a graphical or diagrammatic representation of all the enterprise's business and processes. [Whitman 1997].

2.3.1 Differences between the two Definitions

The above definitions are very similar. They do however differ in that they refer to different aspects and perspectives of an enterprise.

Those who define an enterprise model as a computational representation of its activities, resources etc, refer to attempts made to establish a common platform across an enterprise's boundaries, providing a solution to the communication problem that has been repeatedly reported [Gruber 1996]. The latter definition refers to graphical models of processes in for the purpose of business process re-engineering. We can conclude from the above that there are clear differences between the two definitions. The developments under the first definition try to tackle the communication problem between departments, whereas the latter is referring to business process reengineering models. Both however, primarily examine the functions of the enterprise, build models of the information flow and finally are used to carry out a degree of 'what-if' analysis. The following paragraphs explain the definitions in more details.

2.3.2 E. M. as a Computational Representation

It has been widely reported that there is a lack of communication amongst departments within an enterprise, due to insufficient and incapable infrastructure to support knowledge transfer [Gruniger 1994]. There are clearly social and technical issues for this problem.

Sometimes social and cultural issues such as conflicting interests. between enterprise departments can lead to lack of communication.

Another reason for such a problem is the lack of development standards that would force developers to stick with a particular set of tools regarding knowledge representation and acquisition. Usually when people deal with complex systems they tend to decompose them into small, more manageable parts, and develop these parts in isolation. This is classic management teaching that accompanies system analysis, design and development. Although this approach can be useful as a development guideline, it can also result in systems being developed by different people, at different times, using different tools and methodologies. Due to the lack of development standards and cultural differences between developers or managers, we end up having an enterprise system consisting of *island* departmental sectors and applications.

One can conclude that the links that are supposed to be providing information exchange between departments could be inaccurate. Unless there is a common knowledge base that enables free flow of information across the boundaries of an enterprise's departments, information resources will be limited to the information each department possesses. On top of all that one has to deal with the traditional problems of inconsistencies with the terminology each department uses and duplicated information.

There have been attempts [Gruber 1991] to tackle the problems of communication between departments in an enterprise arising from lack of technical infrastructure. One of them is the TOVE (Toronto Virtual Enterprise) model developed at the University of Toronto [Gruber 1991]. TOVE is an ontology that defines the entire enterprise.

The main objective of TOVE is to provide a dictionary of the activities, resources, processes and data within an enterprise. Details of the project are presented in the next chapter. What developers are trying to achieve in technical terms is to establish communication protocols that will 'read' the information of a knowledge base and restructure the knowledge so that another knowledge base using a different representation structure can read it.

2.3.3 E. M. as a Diagrammatic Representation

There is another definition type that assumes that an enterprise model is a diagrammatic representation of an enterprise's activities and resources. Authors who refer to enterprise modelling using this definition, refer to the several attempts for example TOVE which have been made to create a graphical representation of an enterprise's businesses, in order to assist the process of business reengineering. Business reengineering is the process of identifying all the activities of an enterprise's operation in terms of information, labour, equipment, etc. The aim is to assess whether each of these activities is producing the desired output and at the same time to examine if there is any room for change (i.e. to turn a semi-automated function into a fully automated one).

2.4 Why Are Enterprise Models Important ?

Enterprise models often help to anticipate the effect of different conditions on an enterprise. In order to minimise risk, enterprises make forecasts and experiment with fictional scenarios. They test their systems against different inputs and then analyse the outcome and the performance. Models will do both; sometimes simultaneously.

It has been concluded during this research that the majority of enterprise methodologies neglect external factors. External factors affect an organisation's operations; either positively or negatively. Enterprise models should acknowledge those factors that affect internal operations. A benefit of using models with such capabilities is the ability to prioritise alternatives and actions. The changing strategies of integrators in the automotive industry, for example, will cause changes to the entire supply chain model. Consider that an integrator decides to leave the region in which it currently operates. The departure of the integrator will have a direct impact on its suppliers operations. The suppliers will either have to find some other integrator to supply their products, move to another region or close down. With a suitable enterprise model the amount of change and the effect of the changing strategies on the model can be predicted. Building an enterprise dependency model of the above scenario would enable us to develop a picture of the industry, apply the changes caused by the new formulation of strategies, analyse the affects and finally develop a list of alternative actions for the remaining players on the model.

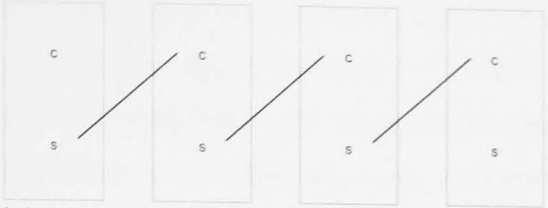

Figure 2.2 one example of a supply chain (c=customer, s=supplier)

Enterprise networks are accompanied by large amounts of information. The amount of information or knowledge an enterprise possesses is usually limited to the enterprise's operations and objectives. Although an enterprise may have some information regarding the industry which it belongs, the knowledge or information they may possess with regard to the operations and objectives of other enterprises in the same industry is certainly limited. Any enterprise in a competitive environment will be reluctant to share with their competitors information of their operations, strategies etc. Models assist in building scenarios of the industry using the data an enterprise has alongside fictional data regarding the operations of other enterprises. The aim to carry out a 'what if' analysis of the enterprise network and reply to questions such as 'what will the impact be on our operations if supplier x leaves the region?' or 'how dependent are suppliers x and y upon our business?'.

Testing ideas and experimenting with fictional scenarios and theories can be expensive and time consuming. On the other hand, it is not possible to wait for something to happen in order to develop alternative procedures. There are, of course, limits to what a model can emulate and enterprise models tend to be very complex structures. There is an enormous amount of information involved and that's why supply chain models tend to focus on one aspect of an enterprise's functions such as inventory control or distribution. This issue addresses two distinct problems. These are namely the problem of representing the dynamics of a model and the problem of representing its scope. Later in the thesis it is shown that these two aspects inter-related and inter-dependent. The challenge of using modelling is to determine the situations that can evolve due to the changing strategies of large organisations that affect the entire industry. The benefits emerge after a model of a situation x has been created, enabling supply chain managers to forecast further situations in case more changes occur.

Finally models help people to visualise complex situations. The objective here is not to convince someone that the situation sketched by the model is the only possible alternative triggered by a series of events. Models are there to help us visualise opportunities. Opportunities that are not visualised are opportunities missed. In a competitive environment, the critical steps are identifying every available opportunity, predicting them before they actually happen, and taking the best advantage of them by taking appropriate action. Models are very advantageous in staying competitive. Unless people are working on data and information handling on a day to day basis, it is unlikely that they will *'see'* all of the possible relationships that exist in a supply chain, or recognize new ones that might emerge. Building a graphical model of a supply chain will help management to see and understand supply/demand relationships, their strength and emerging opportunities.[Mujtaba 1994]

2.5 What is expected from an Enterprise Model

Enterprise modelling is an area that has received great attention in the last few years. Many of the developments in the area are concerned with the communication problems between departmental sectors within an enterprise, which can of course cause inconsistency in the way an enterprise operates. On top of that there is a general tension to moving towards an integrated environment rather than an environment where departmental units are autonomous and work independently of the rest of the enterprise.

The main objectives of a product enterprise at the tactical level are to:

- keep production costs low, cut stock
- improve product quality
- reduce time scales regarding production or distribution.

Whereas the objectives for a service enterprise at the tactical level are to:

- identify customer needs
- perform promotional activities
- cut down on service fees

These are constant challenges that all enterprises face. In order to keep track and monitor the activities that result in faster and better quality production and distribution or service provision, it is necessary to develop a model not only of the individual units responsible for the above actions but also of the relationships and dependencies of these units. The production of a car engine, for example, does not only depend on the speed of the unit that's responsible for assembling the engine but also on all the units responsible for supplying the engine parts from raw material to engine components. In order to monitor the above activities and ensure that the objectives are accomplished, enterprises develop models.

Models usually deal with the distribution or production functions. In the former case they monitor the performance of the marketing department and how fast the final product reaches the customer, whereas in the latter case they monitor the performance of the suppliers and the production plant. The main objective in the first example is to improve quality of the service or product and in the latter to reduce manufacturing costs.

Before one can actually develop a model you need to bear in mind a set of attributes which are associated with enterprise model and supply chain modeling techniques. These are discussed below.

2.6 Attributes of Enterprise Models

One reason for developing enterprise network models is to be able to assess changes that might occur in an enterprise network and their impact on an enterprise's operations.

One of the basic principles that has to be applied in a enterprise model is the ability to monitor market signals across the network and to make consistent forecasts. Historically, enterprises with multiple departments have been making forecasts independently, using their own assumptions, measures and level of detail [Sadeh 1996].

Sadeh suggests that independent self centred forecasting is incompatible with quality supply chain management and proposes collaborative forecasting by continually picking up signals from the market regarding customer demand, ordering patterns and restocking algorithms.

Secondly, the enterprise network model has to be targeted at a specific level of decision making within an enterprise i.e. scope. A single enterprise model cannot support decision making at all three levels; strategic, tactical and operational. However what can be done is to develop three separate models that deal with the three different levels of decision making. The first model would assist with decision making at a strategic level, by modelling the network of enterprises within the same market (supply chain). Another model would deal with short term forecasts, drawn from day to day transactions. Finally a model would assist with planning and production, drawing data from the performance of supplier-manufacturer relationships.

The third objective of an enterprise network model is performance measure. Performance measures will indicate the progress of the model in terms of *'how well is it going ?'* [Artzen 1997]. Supply chain managers take a broader view, adopting measures that apply to every link in the supply chain and include both service and financial metrics. [Anderson 1996]

2.7 How can we make an Enterprise Model Useful ?

Enterprise network models, could be defined as the ones that enable the modeler to create fictional scenarios with regard to the industry that is targeting and make forecasts following a 'what-if' analysis. There are three potential levels within an enterprise that enterprise network models can be useful; strategic, tactical and operational. The major difference is the level of information used as well as information expected by the analysis. The following paragraphs explain what information can be expected by developing an enterprise network model for each of the levels; strategic, tactical and operational.

2.7.1 Strategic

Strategic planning deals with long range resource planning for planning positioning. Often this method analysis is relatively simple [Sadeh 1994]. The results of the modeling procedure will help an enterprise in formulating a strategy and setting up future objectives. The model is expected to assist in making decisions with regard to future investments and exploitation of opportunities within the industry. In order to do this, such

a model must be capable of representing a market's economy and could also be used in restructuring.

2.7.2 Tactical

Incremental adjustments to inventories, storage, capacity, raw materials and transportation need time to arrange [Artzen 1997]. The primary analysis is partly resource planning and partly process planning in the intermediate term. This will help the business respond to changing market conditions.

2.7.3 Operational

Strategic and tactical planning are done on an entire enterprise's capacity or critical resources. Specific products are modeled at the highest level of detail. The intent is to determine the best time to manufacture and schedule production. This is where business process engineering is being applied. Although as we will see in the next chapter business process re-engineering is almost synonymous with enterprise modeling it cannot do anything in changing market conditions or assist to foreseeing future investments and opportunities within a particular industry. It is this stage where business process re-engineering is applied.

Regardless of the level the model is intended to represent, it should carry a set of characteristics in order to successfully reflect the enterprise. The list of criteria presented in the next paragraph can be used as a guideline or for evaluation during the development process. The criteria have been supported by the survey the author carried out. The results are presented in chapter 5.

2.8 How can we make an Enterprise Model Successful?

An enterprise model, regardless of its use, has to carry some characteristics in order to be successful.

- *Completeness:* Before a model can be used to carry out a 'what-if' analysis it has to be accurate. The information has to be up-to-date and the data used have to be cross examined in order to avoid inconsistencies.

• *Scope:* Modelling should start at the strategic level. Enterprise models should not be limited by departmental boundaries.

• *Dynamic:* Whitman [1996] introduced the idea of a model being alive in the sense that it is continually updated with information so that it reflects the enterprise's current status at all times.

• *Expressive power:* The model should be able to represent different levels of detail of the enterprise. This implies the use of different views i.e. consider a generic view of the entire enterprise as opposed to detailed view of a particular process.

• *Open:* Enterprises are open systems with links to the world; data is been passed into the system and given out to the world as output.

Although most of the efforts so far follow the above criteria, the dynamic criterion is often omitted. In the following paragraph we discuss the importance of the final criteria and the work which has been carried out by the author. Bearing in mind what has been said so far about enterprise modelling, the author will show in the next few paragraphs why we need to animate enterprise models. We examine the nature of enterprise dependencies and the changing roles of businesses and their responsibilities.

2.9 The need for dynamic modelling

It has been concluded during this research that static enterprise modelling cannot successfully reflect the state of an enterprise as part of a wider market or a supply chain due to the vast amount of changes in business roles, responsibilities, technological achievements and the global nature of work. An observation that was made during the survey of the literature review of this thesis is that modelling in general has three dimensions that are scope, dynamics and informatics. It has also been concluded that the scope of any modelling technique is dependent on its dynamics. For example enterprise modelling techniques that cover a wider set of enterprise aspects tend to be less flexible (or static) than other which target a particular aspect of it. So why do we need dynamic models after all.

It has been observed [Evans 1996] that in the last few years huge investments have been made for the improvement of legacy information systems. These investments target technological or structural change and therefore aim to successfully perform tasks such as down-sizing, right-sizing, reverse engineering, re-engineering, client - server architecture etc. Although most of these projects end up successfully, the total efficiency of the entire organisation did not increase dramatically. This happened because the IT projects were carried out without reengineering of the business processes, including both manual and automated tasks. The changing of process re-engineering is sometimes proportional to the technological pace and the distribution of responsibilities in a market or supply chain. In the following paragraphs, the author discusses the nature of enterprise dependencies, the role of modelling tools and their abilities to model successfully an enterprise's environment, in an attempt to make a case for dynamic modelling, which is in fact the basis of this thesis.

2.10 Modelling Enterprise Dependencies

The enterprise network modelling approaches that were reviewed during this research, (Chapter 3 discusses the approaches in detail) consider enterprises as autonomous entities. Their goals and achievement lie primarily on the performance of all of the key tasks. One has to bear in mind that enterprises (regardless of their market position) are not closed systems. They are open systems with links to the outside world and they heavily depend on external input. As Figure 2.3 suggests, these links or dependencies do exist, and play an important role in the enterprise's success.

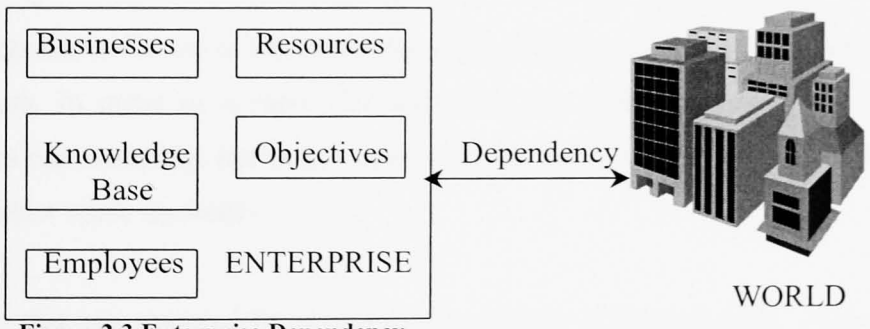

Figure 2.3 Enterprise Dependency

Consider, for example, the automotive industry. In order for car manufacturer A to produce a pre determined X amount of cars every month it has to ensure not only that all the businesses of the enterprise work 100%, but also that the links with the outside world are continually providing the company with input.

If one tries to draw a diagram of the process of building a car, one would end up with a hierarchy, where the bottom level is raw material and the top is the car. When, for example, an electronics supply company has a product capacity of 10000 items a month and is hoping to carry on at the same rate, it really depends on the layer below in the hierarchy. Regardless of the company's performance if the layer directly below malfunctions then this will have a direct impact on the company's operations. The same conditions apply to the entire supply chain. If an event like the one just described occurs in the automotive model, then the entire model will be affected in some kind of direct or indirect form.

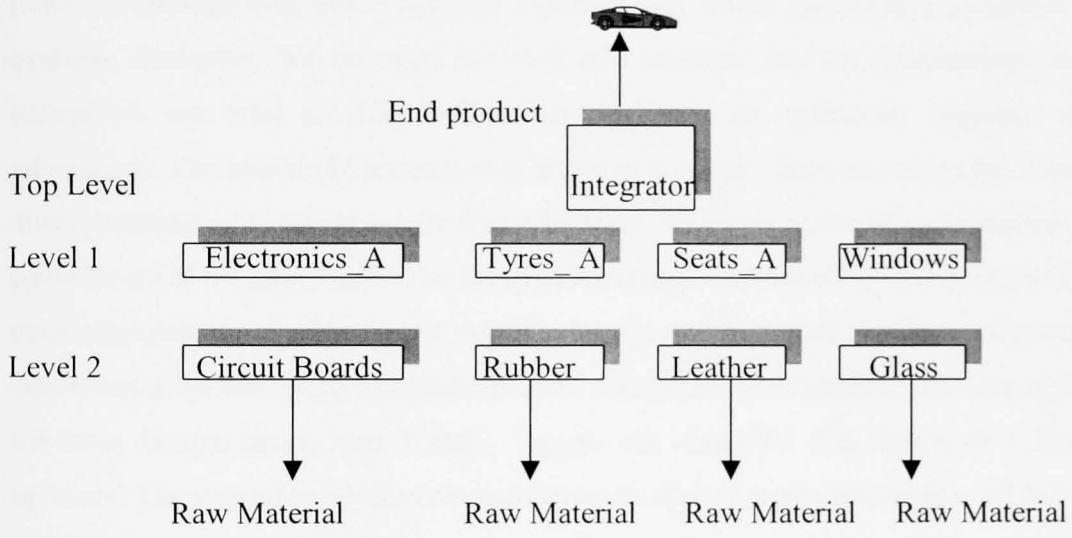

Figure 2.4 Hierarchy of Dependencies

The integrator at the top of the hierarchy is an enterprise with a product capacity of X cars per month. In order to achieve that goal first the enterprise has to ensure that all its functions run smoothly, but at the same time the functions of the other companies at the levels below work smoothly.

Consider that raw material for the Level 2 company *'Circuit Boards'*, cannot be supplied for one month. This will have a direct impact on the company "Circuit Boards", but it will also have an indirect effect on the level above.

If raw material cannot be provided at Level 2, the Level 2 companies won't be able to supply Level 1 until raw material supply is resumed. Therefore the Level 1 company

'Electronics_A' will not be able to produce, and the integrator at the top will miss its electronics supply, unless other electronics suppliers can increase their capacity and bridge the gap. Assuming there aren't any other electronics suppliers, the integrator may be forced to lower the production of cars temporarily. This will have an effect on Level 1 as the integrator will require less supply until the situation is recovered, and Level 1 consequently will have to lower productivity too.

One can see how one malfunction at the lowest level of the hierarchy can have an impact all the way up to the top level. The lack of raw material at level 2 had an affect throughout the model affecting the operations of the entire supply chain. Even enterprises that are not directly related can affect each other. The electronics and tyres supplier of Level 1, although they are in different business they affect each other's operation as the example illustrates. We can also see from this example that the relationships between enterprises are vital to their success in all levels of operation; strategic, tactical operational. The above observation was made in a supply chain environment. Links and interdependencies however can be found between the sub-systems of an enterprise or the components of a system itself. The thesis concentrates on animating relationships between interconnected agents. By agents one can imply anything from system components to departmental sectors of an enterprise or even enterprises in a supply chain. The animation has been broken down into 3 steps. Agents are classified and stored in a thesaurus structure. The thesaurus enables the development of an initial static picture of the model. The PAD rules that are explained in chapter 4, show how the model can move in terms of relationships added or deleted. Finally lisp type lists are used to move as well as keep track of movement. The consequence of moving around responsibilities using an algorithmic approach, like the one discussed in chapter 4, is that it makes essential an additional mechanism that determines whether the new share of responsibilities can be found in a real life situation. Chapter 5 raises this issue while chapter 4 discusses the method for animating models. Before going to the literature review we take a look at the role of modelling tools in general.

2.11 The role of Modelling Tools

[Butler 1999] observes that the world of Computer Aided Software Engineering (CASE) tools appears to be turning full circle. Modelling has been back in vogue for a while, and

now integration is returning. Whilst CASE became a victim of bad publicity, in truth, little of it was to do with the capability of the tools, which demonstrated that modelling was more than just pictures. CASE ensures the consistency and completeness of diagrams, and enables activities that span the Systems Development Life Cycle (SDLC) to be integrated together with the work of many developers.

However, whilst a diagram has always been understood to be worth a thousand lines of code, the problem of the last decade has become 'which diagram?' Development methods have been just as much a victim of fashion as any technology. Last year we could be forgiven for expecting the Unified Modelling Language (UML) diagrams to become the dominant modelling technique, as it was the first to become anything close to a standard following its adoption by the Object Management Group (OMG). Yet, UML falls short of meeting all of an organisation's modelling requirements and, of course, Object Orientation (OO) has not become the universal paradigm for which its supporters had hoped. Those who are leaning towards UML will still have a legacy of models from previous approaches they will want to reuse.

As a consequence we see a diverse range of different methods in place in most organisations, with point solutions often used to address different pieces of the puzzle. This results in little opportunity to exchange modelling information across projects, or across the SDLC, with an ensuing lack of consistency. It also results in many developers having a whole array of tools not just to support their own modelling needs, but simply to reuse the results of others. The lack of integration also introduces errors into the process, as there is little way of ensuring that one diagram is a proper reflection of another.

2.12 This Research

The aim of this system is to produce a modelling methodology that produces enterprise models that can be animated. Animation of enterprise models is a difficult task because it involves a great number of parameters that has to be taken into consideration. On top of this, the author has concluded by reviewing previous attempts, that the scope of the model is proportional to its dynamics. That means, that the wider the scope or the more aspects we model the less flexible the model becomes. One can compare this with a simple C programming function. The more parameters are being passed into the functions the more

if conditions will have to be performed in order to assess the parameter's output result. Something very similar is happening with enterprise models the more aspects we are targeting such as logistics, services, ownership, resources, responsibilities, the more complex the model becomes.

This target of this thesis is the development of a method where models such as the ones examined in paragraph 2.10 can be developed and animated. The approach also allows us to assess the impact of the animation on the entire model. Of course the changes are not always of the nature described above and certainly do not have the same impact. The proposed model will assess the impact of an agent entering into or leaving the model. The issues raised as well as the strategy proposed are described in Chapter 4 and 6.

2.13 Conclusion

The chapter gave a clear definition of the term enterprise network modelling and distinguished this term from the general enterprise modelling definitions. Section 2.4 and 2.5 discuss features of general enterprise models in terms of scope. The author draws attention upon those issues that current enterprise models omit and discusses the need for an enterprise network model that enables the modelling enterprise dependencies. The principles discussed in section 2.7 and 2.8 would assist in reducing the error margin during forecasts and hence enhance decision making. Clearly there is a line that has to be drawn during the development of an enterprise model. The line will be derived by assessing the purpose of the enterprise model, and the level of support it is intended for.

The example in the section 2.8 showed that there is a need to take a wider view in order to build an enterprise model that can carry out a successful 'what if' analysis. The implication of expanding the boundaries of the model in terms of information are undoubtedly vast. However, when moving towards a global market this wider view becomes necessary and we are constantly moving towards the era where such a model becomes essential.

Chapter 3 Literature Review

3.0 Introduction

The chapter reports on the wide range of modelling methodologies and products offered for enterprise modelling. Each product or methodology concentrates on different aspects of the enterprise ranging from I/O models to process modelling, product manufacturing or product/service distribution. The chapter first distinguishes between architectures, tools and methods and gives an overview of the terms associated with enterprise modelling such as business process re-engineering, enterprise engineering and ontological modelling. The overview will help the reader to distinguish between the different aspects of enterprise modelling targeted by the various methodologies and tools. Later on in the chapter it is reviewed in detail the methodologies which have been mostly relevant as well as influential to this research, emphasising at the same time the need for model dynamics in all three levels of the enterprise. The methodologies have been divided into two broad categories, distinguishing between research and commercial products. Most of them are associated with business process re-engineering and the final model is static. There are however modelling techniques aiming to run scenarios and experiments that offer facilities for model animation. A general observation that could be made about static as opposed to dynamic techniques is that static techniques tend to consider a wider view of the enterprise. Dynamic techniques target specific aspects such as logistics, sales, distribution etc. Static techniques on the other hand do in fact model a wider range of aspects because dynamic techniques have not yet been developed enough.

The same observation can be made for commercial products. Some of them span their scope throughout a supply chain targeting the modelling of logistics such as freight control while others target enterprise modelling aspect such as improvement of processes from an operational to strategic level.

3.1 The Modelling Process

The process of modelling is to create a general model or an abstraction of the real world. This model reflects characteristics displayed in the real world model, to a

chosen degree of detail [Kochikar 1994]. Models focus on essential characteristics only, filtering out information that is irrelevant or superfluous to the task at hand. The typical approach to modelling requires initially an identification of the model. This is done by the modeller who decides which system characteristics are related to the purpose and then constructs a general structure which is flexible enough to allow scope and additional complexity, enabling detection and consolidation of components and their interactions. The type of modelling described above is purpose driven and tool dependent and it has the inherent disadvantage that different models have to be constructed for different purposes despite the fact that it is the same system being represented. Although this is a widely known way of managing complexity, it has been suggested [Duse 1992], that the problem can be overcome by building a reliable model which is tool independent and that is able to represent manufacturing enterprises from which the relevant information can be extracted. This model is known as a base model, it consists of complex and extended representation of a system and it is a composite of physical, control, and informational elements as well as interactions between these elements.

The concept described above is a new paradigm in manufacturing systems modelling, however more research is necessary before this concept can be fully utilised. The design or analysis of a system is a very complex task due to various factors such as cost, quality, flexibility, time, and size constraints, and thus requires good modelling methodology and tools. To understand the complexity of the system at a manageable level requires the use of abstract or general models and well-defined architectures. [Little 1991] proposes that due to 'bounded rationality' humans prefer to break down complex units into components for purposes of analysis, management, and design, this idea developed from systems theory. Different aspects of the system have resulted in distinct models due to the preference of humans to small, manageable, abstract pieces of reality. It is common knowledge that every system, with few exceptions, is a composite of subsystems and these components or subsystems are more often conceptualised as whole systems in their own right. Such subsystems are frequently modelled separately and thus may make a holistic view difficult and there may also be difficulties in moving from one model to another [Savolainen 1995]. The result of utilising these tools is a model such as a database, and methodologies is the term used to describe the process used to make a model accomplish a desired effect, what is

accomplished can vary, be it a higher level model or an architecture. The following sections will provide an overview of existing modelling issues, and distinguish between existing architectures, modelling tools, and modelling methods. Some of the tools and methods presented below have been more influential than others to this research and are therefore discussed in more detail later in the chapter.

3.2 Enterprise Architectures

The architectures used in CIM systems include conceptual models and regulations that enable translation of the model into a working reality. [O'Sullivan 1994] proposes that architecture is; "*a body of rules that define those system features which directly affect the manufacturing environment into which the system is placed. These features include system configuration, component locations, interfaces between the system and its environment, and mode of operation*". There are two well-known types of architecture; a reference architecture and a particular architecture. The former refers to a detailed collection of common attributes management and automatic control tasks and the functional necessities. The latter is the instantiation of a reference architecture.

The CIM reference model committee of Purdue University [Tham 1995] describes a reference model architecture as a formally agreed upon standard definitive document or a conceptual representation of a system, while independent of any specific requirements of any particular task the reference model recognises common essentials required for all implementation. Essential modelling principles applicable to enterprise models include [Vernadat, 1996]:

- The *principle of separation of concerns* to analyse a business entity piece by piece and thus break down complexity of enterprise models;
- The *principle of functional decomposition* introduced by SADT and based on a stepwise-refinement approach to go from a general view to detailed views of a system;
- The *principle of modularity* where models are made of interconnected building blocks, thus facilitating management of change and maintenance of models;

- The *principle of model genericity* where standard building blocks are defined as generic classes which can then be specialised to particular needs;
- The *principle of reusability* to reduce modelling efforts and increase model modularity by the use of pre-defined partial models;
- The *principle of separation of enterprise behaviour and functionality* to increase organisational flexibility, i.e. enterprise behaviour can be updated without changing installed functionalities and vice versa;
- Finally, the *principle of process and resource de-coupling* which structures an enterprise as:

1. a federation of agents called functional *entities*; and
2. a large collection of communicating business processes processing enterprise *entities* and synchronised by the exchange of events and messages.

The last principle is very important in order to get the operational flexibility of the organisation. Business processes indicate the overall logic of what has to be done in the enterprise to achieve business objectives. Functional entities are autonomous units (devices, applications or people) which must execute business processes according to real-world situations, imposed due dates and must react to perturbations. If resources are tightly coupled to processes, the model defines a rigid structure not suitable for the agile enterprise. On the basis of these requirements, enterprise reference architectures have been defined such as CIMOSA or GERAM. All these architectures recommend that an enterprise modelling paradigm should deal with:
- three modelling levels for requirements definition, design specification and implementation description of the business entity;
- at least four modelling views, namely:
- function view to cover enterprise functionality and behaviour aspects
- information view to cover data and information aspects
- decision/organisation view to cover decision centres and responsibility aspects
- resource view to cover resource requirements and management aspects
- the role and place of humans in a particular enterprise architecture
- and be supported by a modelling methodology.

Numerous reference architectures that have been proposed by many individuals, computer manufacturers, and collaborative research projects deal with the principles mentioned earlier. The most important that are frequently mentioned in this thesis are discussed briefly below.

3.2.1 CIM-OSA

CIM-OSA is an open-system architecture that defines an integrated methodology to support all phases of CIM system life cycle, from requirement specification through to system design, implementation, operation and maintenance, and even system migration towards the CIM-OSA solution. The system designer is guided by his architecture into deciding what should be implemented to achieve the required end. Described using formal terminology it can be described as deciding on an implementation description from requirements definition via a design specification. This particular situation requires a model developed from the methodologies used for addressing the generic requirements. During each stage it is stressed, [Jorysz 1990], that it is important for the designer to focus his attention to functional, informational, resource, and organisational aspects of a CIM enterprise.

3.2.2 ICAM Architecture.

ICAM or (integrated computer aided manufacturing) has developed an architecture using tools such as IDEF0 (ICAM DEFinition – Zero), and IDEf1 (ICAM DEFinition – One). The architecture discussed is defined as hierarchical decomposition or top-down approach. The architectures developed by CIM are not generally available to the public [O'Sullivan 1994].

3.2.3 CAM-1 the Architecture.

The Computer aided manufacturing - international (CAM-1) architecture is an abstract model of manufacturing enterprises. It was created using the method of functional decomposition which permits important details to be displayed such as company policies and procedures, organisational structure and standards.

3.2.4 NBS Architecture:

The National Bureau of Standards or NBS, their architecture uses a hierarchical control approach consisting of five levels of hierarchy structure such as factory, a

shop, cell, workstation and machines. Each level or system can be broken down further into smaller components. The decomposition is based on procedures. functions, or regulations providing a line of a lower level commands. This architecture was developed to enable the manufacturing system vendors to create products compatible with the CIM [O'Sullivan, 1994].

3.2.5 IMPACS Architecture:

This architecture uses IDEF0, data flow diagrams (DFD), (GRAI) grids and nets, IDEF1x and group technology. IMPACS is not the first attempt to bridge the gap between global planning and production control strategic planning around the real time control at cell level [Doumeingts 1995]. Software modules such as dispatcher, scheduler, mover, producer, dispatcher, and monitor are used to control the production cells. All of the above mentioned software modules are designed to be compatible even if they are developed by different vendors. It has been proposed [O'Sullivan 1994] that the IMPACS architecture is widely accepted among manufacturing software vendors as a practical interpretation off the production management system.

3.3 Modelling Tools

All models are built using modelling tools in the process described by modelling methodologies, The term system modelling tools referred to techniques used to diagrammatically represent that functions or activities [O'Sullivan, 1994].

3.3.1 IDEF Modelling Tools:

IDEF0, IDEF , IDEFx, and IDEF2 are all modelling tools developed by ICAM project of the US Air Force at the Soft Tech Inc. These three central modelling tools which are complementary to one another, provide a functional, informational, and dynamic models of the system. IDEF models are used mainly for requirement definition and thus IDEF0 functional models are not able to interconnected with IDEF1 informational models, and IDEFx are suitable for building semantic database models.

3.3.2 IDEF0 Models:

This is a highly comprehensive modelling language that is able to graphically represent innumerable types of business, manufacturing and other types enterprise operations to the desired level of detail required. There are three important features of IDEF0 approach, which are activity modelling graphics, gradual exposition of details and a disciplined teamwork. The basic unit of a IDEF0 model is a function block which is linked to other function blocks through inputs, outputs, mechanisms and controls. Links between the blocks exist in the form of physical objects such as material, and information flow. [Pandya 1995] identifies three essential features of IDEF0 models which are context (position the subject model takes up in the systems hierarchy), viewpoint (perspective adopted by the model), and purpose (reason the model exists). IDEF0 has been widely used and is well thought of in the industry due to numerous factors including; the ease with which it can be used, computer support and conciseness as well as documented rules and processes. A major drawback of this tool is the staticness or inflexibility of the models produced.

3.3.3 IDEF1 and IDEF1x Models:

This technique of modelling is to model the structure of information in order to discover the information requirements of the function, an simple example of this is; "What information should an invoice possess?" This technique is developed from the work of [Chen 1976] on the entity-relationship model. IDEFx is an extension of IDEF1 and is responsible for the flow of information and it provides information on the models of the system which distinguish the structure of the information required to support the functions identified using IDEF0. IDEFx is not widely used due to the inability to support for composite entity types and the rigid methods required to use it such as a complete counting of characteristics of entities before instantiation [Pandya, 1995].

3.3.4 IDEF2 Models:

This model (IDEF2) is an unsupported simulation language however other simulation languages such as ARENA and PROMODEL or more frequently used.

3.3.5 Structured System Analysis (SSA):

The Structured system analysis is a flexible modelling tool, which can be used either for modelling data flows or modelling the flow of physical units. It is very similar to IDEFO in the fact that it utilises principles of hierarchical decomposition and modularising functions. It has been proposed [O'Sullivan 1994] that SSA is more detailed and orientated to software than IDEF0 and is the most well used modelling tool for data flows.

3.3.6 GRAI Grids and GRAI Nets:

The GRAI models numerous activities with reference to decisions and information flows between these activities while the GRAI net models and highlights the decision making process itself. These tools were produced by the GRAI laboratories in France, in order to model decision-making processes that take place in manufacturing environments.

3.4 Modelling Methods

These methods are used a to provide up basic outline of how modelling tools can be integrated to model a specific system. There are many modelling methodologies for modelling CIM enterprise systems including IDEF, SSADM, SADT, GIM, CIM-OSA cube, MOOD, and the M approach, each of which will be discussed below.

3.4.1 IDEF Method:

This method was developed to enable designers to answer three essential questions with reference to manufacturing systems; activities performed, information and data requirements of each function, and the changes that occur over time. Three language tools were developed in response to model all three aspects, these language tools were IDEF0, IDEFx, and IDEF2 and these tools were used to develop ICAM architecture.

3.4.2 Structured Systems Analysis and Design Methodology (SSADM):

SSADM was especially developed to be used in system development projects, and is a procedural framework which uses three modelling tools: data flow diagrams, and logical date structures, and entity life histories to provide function, data, and event views of the systems [Pandya 1995].

3.4.3 Structured Analysis and Design Technique (SADT):

[Ross 1997] puts forward that SADT is exactly what the title infers: a structured analysis and design technique. This tool makes use of various graphical and textual tools including activity diagrams, data diagrams, node lists, and data dictionaries to model the structure of the system in question, using a top down approach. It is important for the analyst to focus upon the activity and data views of the system being modelled and encourage integration of systems. A functional model is essential for the methodology before a physical design is possible. This methodology is supported by software packages such as AUTOIDEFO and SPECIFIX.

3.4.4 GRAI Integrated Method (GIM)

The GRAI Integrated Method (GIM) [GRAI 1993] consists of the integrated use of the tools IDEF0, IDEFx and Group Technology. The GRAI is used by the ESPIRIT IMPACS project that utilises an integrated approach.

3.4.5 CIM-OSA Method:

This method [CIMOSA 1994] combines many well accepted concepts and principles used in modelling tools, such as functional decomposition (SADT), function/activity and informational modelling (IDEF), the entity relationship model. The three-schemer approach utilised in ANSI/SPARC development in data communications and computer networks (MAP, OSI, TOP) and integrated manufacturing modelling performed by IBM, NBS and CAM-1.

3.4.6 Object-Oriented Approach:

This approach although relevant to this thesis, is only discussed very briefly here. This approach provides a new ontology for enterprise and CIM modelling. Many authors [Ngwenyama 1994], [Kim 1993] proposed an object-oriented approach to modelling CIM information systems. Kim proposes a methodology which consists of two phases; an analysis phase which decomposes component functions of manufacturing in order to define the information flow among the manufacturing functions and their infrastructures, and secondly a design phase. In order to describe manufacturing functions, functional diagrams are used, these diagrams are transformed into and object-oriented information model consisting of a class

dictionary and class relationship diagrams. The class dictionaries can then be translated to a specific data dictionary in an object-oriented database management system.

3.5 Business Process Re-engineering

In the general sense business reengineering is the process of identifying all the elements of an enterprise's functions in terms of information needed, labour, equipment etc, and then carry out an assessment on the performance rate of the function. The aim is to assess whether each business function is producing the desired output and at the same time examine if there is any room for change e.g. by turning a semi-automated function to a fully automated one. The term business process re-engineering (BPR) is often found in the context of enterprise modelling. In fact a great majority of enterprise modelling tools is dedicated to BPR. At this point one may ask what is exactly BPR? Although many definitions have been proposed, the majority provides only vague approaches. Even the term "reengineering" is something of a misnomer. It suggests that the business process was initially engineered at its inception [Morris 1993]. We will now make our own attempt at clarifying the terminology relevant to BPR.

[Hammer 1993] defines Business Process Reengineering as "the fundamental rethinking and radical redesign of business processes to achieve dramatic improvements in critical, contemporary measures of performance, such as cost, quality, service, and speed." Continuous Process Improvement (CPI) is the collection of activities that are systematically and continuously performed to bring about enhancements in enterprise performance. The main difference between BPR and CPI is in the extent of improvements targeted by these two methodologies. BPR targets radical change while CPI is focused on incremental change [Mayer 1998]. A related methodology, Total Quality Management (TQM) is "a means of operating a business that seeks to maximise a firm's value through maximising customer satisfaction at the lowest possible cost" [Spitzer 1993]. Therefore, TQM is the systematic application of methods and tools to accomplish CPI.

One of the predominant distinguishing characteristics of engineering that separates it from other professions is the creation and use of models. Whether these be physical models, mathematical models, computer models, or structural models; engineers build and analyse models to predict the performance of designs or to understand the behaviour of devices [Mayer 1998]. In the context of BPR then, we use the term "engineering" to imply the development of a process system with predictable behaviour using some methodology that employs models as a basic tool. Whether that new system requires re-structuring or automating existing processes the engineering element must provide the modelling support required. Evolutionary techniques use models that enable quantitative analysis of a proposed change to a process. Some of these techniques are examined in this chapter. Continuous improvement using such models can achieve the lowest cost, highest performance implementation of a process whereas breakthrough change of a process (paradigm shifts) generally is preceded by establishing a shared understanding of the fundamental nature of the situation at hand[Mayer 1998].

3.5.1 The BPR Process

This section provides a brief description of the BPR process. BPR is reportedly complex and requires a considerable amount of effort of personnel with many different kinds of skills and experience. Successful re-engineering requires a team that is committed to the accomplishment of the processes objectives. Of course one can go deeper into the area of BPR but this is beyond the scope of this thesis, although there are a number of references in the bibliography section. The methodologies examined in this chapter focus on the task aspects of activities that are important for the success of BPR projects, as well as the interrelationships between these activities. Business re-engineering activities involve many different cognitive tasks such as conceptual design and analysis, detailed design and analysis. The description provided in this chapter is intended to offer an insight into the activities of BPR methodologies.

3.5.2 Definition BPR Vision, Mission, and Goals

An essential, and early step in a BPR project is to clearly understand the mission of the organisation implementing and define a vision for the re-engineering effort that is consistent with this mission. An enterprise mission statement is a statement about an enterprise that summarises the reasons for its very existence. For example, the mission

statement of a Widget manufacturing company may be "To be world-wide market leaders in the Widget business." It is important to be cognisant of the enterprise mission while developing the vision of a BPR effort. A statement of the vision of a BPR effort may be " To increase the profit margin of the XYZ Company by 100% over the next two years." Vision definitions typically cover long periods in time and are at a coarse level of granularity (that is, they are not very detailed). BPR envisionment is usually done by top-management executives, and lead to the definition of BPR goals and objectives. The goal(s) describe the desired outcome(s) of a BPR project. The objectives are a more detailed description of the goals.

3.6 Enterprise Engineering

In today's enterprises, knowledge and information are key resources on a par with capital, personnel, equipment and plant. Information systems are tightly interwoven within today's enterprises, requiring close coordination between information systems professionals and business engineers. The impact of business engineering requires significant change in how information is processed in an enterprise, which means that systems supporting changed processes must also be changed. Along with business processes re-engineering, enterprise engineering is term widely used in this thesis. It is in fact a branch of requirements engineering which deals with the very early stage of system design.

Enterprise Engineering, which applies equally to well-established and newly-formed enterprises, responds to the fundamental business drivers of the 1990's: migration to "agile" production, globalisation of markets, changing labour pools, and volatile political and business environments. The basic element of successful Enterprise Engineering is the linkage of all critical elements. Enterprise Engineering methodologies and tools allow an enterprise to define its strategy, then design and implement processes which support the strategy, and then manage the processes to assure enterprise success -- all while maintaining focus on goals, success factors and stakeholder expectations. Linking information technology to business contribution is an important first step in supporting enterprise performance improvement.

In the area of information system analysis, for example, instead of concentrating on some implementation aspects such as data and processing in a specific application, overall enterprise specification is developed. Its purpose is very similar to other enterprise modelling methodologies, in the sense that they all try to identify problems with current architectures and introduce or suggest alternative designs.

According to [Gustas 1998] enterprise engineering can be divided into five stages. The first stage is concerned with systems analysis of the enterprise current situation (the "as-is" model). The deliverable of this stage presents a description of the present situation of the enterprise including business processes, objectives and common problems. The second stage is concerned with enterprise modelling and integration. It aims to integrate the various models that resulted in phase one. Enterprise Engineering provides both a road map and a vehicle for an enterprise's journey into the future. The Enterprise Engineering life cycle involves a multi-phased approach that coordinates strategic, operational, and organizational demands. The following is a typical Enterprise Engineering cycle:

1. Describe the enterprise mission in a brief statement of purpose: what the enterprise does, how, and for whom.
2. Make assumptions and gather data about external factors; for example, government policies, rates of inflation, markets, and demographic changes.
3. Assess enterprise strengths and weaknesses.
4. Establish goals and objectives and measures linked to the enterprise mission.
5. Develop strategic and operational plans to meet the goals and objectives.
6. Design/re-design and integrate cross-functional processes to meet goals and objectives.
7. Implement information systems that support enterprise processes and assist decision-making.
8. Evaluate performance to ensure that goals and objectives are being met.
9. Re-evaluate and change goals, objectives, processes and measures as necessary.

Enterprise Engineering often involves wholesale enterprise culture change, and is quite difficult. The innovative and constructive use of computer-based tools, at every step in the cycle, can make such change much easier.

Enterprise planning and change analysis is a precise description of actual problems and goals. According to the goals of the business processes a description of appropriate changes is being specified. Enterprise business processing engineering dominates the fourth stage. This stage results in the description of the enterprise at the new desired situation (the "to-be" model). The final stage is enterprise assessment exercise. At this point the new situation as well as the overall methodology is being assessed. The results are validated against techniques such as risk management and quality assurance. Overall enterprise engineering or enterprise re-engineering serves as an alternative view to system development. [Gustas 1998] claims that the strength of the process lies in the fact that several aspects of the information system are being captured and therefore they are directly applicable to the development of the desired information system.

3.7 Ontologies for Enterprise modelling

[Gruninger et al 1996] of the University of Toronto attempted to tackle the problem by creating an *Ontology* for enterprises. *Ontology* is a new concept in this context and in order to understand the work that was carried out by the Toronto team it is necessary to first explain the meaning of the term. [Gruber 1993] defines an ontology as a specification of a conceptualisation. Ontologies were developed to enable knowledge sharing. The concept of knowledge sharing and re-use can be easily understood if one thinks of the requirements necessary to start a conversion or a dialogue with someone else. Both parties have to agree on the communication protocols prior to the commencement of the dialogue and then obey the rules set by these protocols. Language is likely to be the major issue. Establishing a common language as the basis for a dialogue will ensure that a two-directional conversation could begin, where the two parties can exchange ideas, derive a solution and finally solve problems. Another issue of major importance, is the terminology that the two parties will use. Terminology refers to those special and unique symbols or words that people may use when they exchange information. Symbols or keywords can be used to mean different things or represent different data. This may lead to misunderstandings or inconsistencies. Terminology agreement will ensure that the

two parties are referring to the same things when they use symbols, commands or special keywords.

Imagine two people trying to communicate in different languages with no knowledge of a common language which can be used as a basis for information exchange. At this point the parties are isolated and the knowledge they possess cannot be re-used because of the language barrier. The same applies to departments that have no means of exchanging knowledge. This problem can be solved by using a human translator, who listens to one speaker, translates the information from the speaker's language to his own and then translates this into the second party's language. Figure 3.1 illustrates the process of translation during a conversation between an Italian speaker and a French speaker. The conversation is assisted by a third party who speaks French and Italian. The human translator listens to what the Italian speaker says and translates it to French. At the same time he listens to the French speaker and translates back to Italian. The arrows on the diagram represent input and output received and send to and from each speaker.

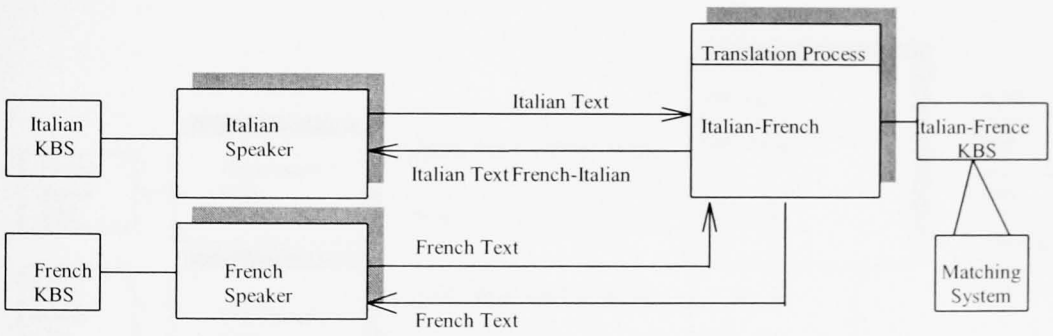

Figure 3.1: The translation process

Assume that each of the speakers has knowledge about his native language stored in some knowledge base. As the diagram suggests the translator would require the same information as well as a system for matching each Italian word or phrase to the French equivalent. Although this process may seem straight forward, in computing terms it can be complex. Knowledge bases are often developed using different methodologies or representations. The functions or algorithms that process this knowledge are then derived from these structures. In order to share some domain of

common interest between knowledge bases we use *ontologies*. As [Uschold 1996] described:

'An ontology necessarily entails some sort of world view with respect to a given domain. The world view is often conceived as a set of concepts (e.g. entities, attributes, processes), their definition and their inter-relationships; this is referred to as conceptualization, which may be explicit -existing in one's head- or embodied in a piece of software.'

An ontology includes a vocabulary of the terms and some specification of their meaning. The ways the vocabulary is created varies. Some vocabularies are highly informal. Others are semi-informal, and expressed in a restricted and structured natural language form using symbols and other notations. There are also special formed languages developed specifically for the development of ontologies. *Ontolingua* [Gruber 1992] was created for the development of ontologies in enterprise modelling. Figure 3.2 shows how two departments with different knowledge structures can communicate via an *ontology*.

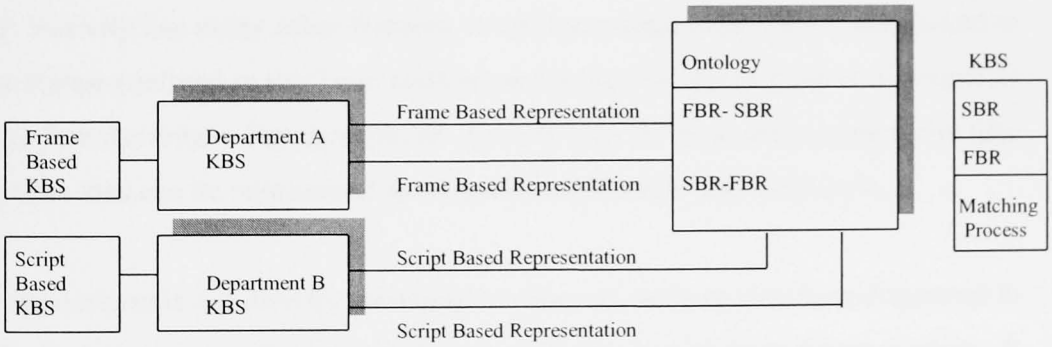

Figure 3.2: Ontology for knowledge sharing

The purpose of every ontology related to enterprise modelling is to support integration within the boundaries of the enterprise by making available a common knowledge representation. This maximises the communication potential and, on the other hand, minimises ambiguity and misunderstanding. The Artificial department of the University of Edinburgh has been working towards the development of an enterprise ontology which defines all the terms related to an enterprise. The following paragraph discusses the enterprise ontology effort in more detail in the context of the enterprise.

3.7.1 Enterprise Ontology

The 'Enterprise Ontology' [Uschold 1998] is a project developed by Artificial Intelligence Applications Institute at the University of Edinburgh with its partners: IBM, Lloyd's Register, Logica UK Limited, and Unilever and supported by the UK's Department of Trade and Industry under the Intelligent Systems Integration Programme. The 'Enterprise Ontology' is a cluster of terms and definitions relevant to business enterprises. The 'Enterprise Ontology' is conceptually divided into a number of main sections which are summarised below.

3.7.1.1 Activities - Processes

The key term is Activity, which embodies the notion of anything involving doing, especially pertaining to the concept of action. The concept of Activity is closely inter-twined with the notion of the Doer. The Doer is a term which could imply a Person, Organisational-Unit or Machine. These terms are collectively referred to as Potential-Actors, that are imbued with the Capability to be the Doer of an activity or skill. These actors may also have other Roles relating to Activity such as Activity-Owner. All of the above terms are defined in the Organisation section. Activity is intimately connected to Resource, due to the fact that the latter is used or consumed by the former. Activity has many other features, it can have outputs or effects, it is linked to a Time-Range (defined in the Time section) and it may be represented as a composite of many Sub-Activity's. For example an Activity may be large and complex and take a long time, this can be represented as composition of many Sub-Activity s.

The term Activity is not directly correlated to time, an Activity may have happened in the past, be happening in the present or may refer to a hypothetical future Activity. It is important to refer explicitly to specifications or plans for Activities. This is encapsulated by the term Activity-Spec. An Activity-Spec pinpoints at some level of detail one or more possible Activities. An Activity-Spec that has an Intended-Purpose is called a Plan. While the term Process-Specification denotes the repeatability of an Activity or Plan. An important aspect for enterprises is the Command of doing of Activities, this is supplied by the Relationship Hold-Authority which denotes that an Actor has the privilege to perform the Activities specified in an Activity-Spec.

3.7.1.2 Organisation

The Organisation section has two central concepts: Legal-Entity and Organisational Unit (OU). Both can refer to either an individual or composite and both refer to things possessing a 'gestalt'. Despite these similarities they differ in a crucial way. Legal-Entity has rights and responsibilities, including legal jurisdictions, which extend to the world at large while OUs require full recognition only within an organisation. The term Legal-Entity may refer both to a Person and Corporation and larger Legal-Entities may completely own smaller Legal-Entities. An Organisational-Unit can be large and complex and may even transcend Legal-Entities. These large OUs will usually be a composite, made up from smaller OUs. In fact the smallest OU may be just a single Person, and a particular Person could be perceived as corresponding with more than one small OU. Roles that are normally played by a person or an OU (such as perform an activity) may be occupied by a Machine, which is by its very nature a non-human, non- Legal-Entity.

From a legal standpoint the Ownership of rights and responsibilities must lie with a Legal-Entity. However rights and responsibilities for Resources, within an organisation, may be allocated to OUs and OUs may sometimes be responsible for Activities. The preceding example is taken into account through the definition of the term Ownership and a distinction is drawn between Legal-Ownership and Non-Legal-Ownership. The management structure within an organisation is represented by Management Links. The term Manage is used to refers to assigning Purposes to OUs. A pattern of Management Links is used to define an Organisational Structure between OUs. This can include multiple Management Links into any one OU with constraints on the different type of Purposes assigned through each link.

3.7.1.3 Strategy

The keyword for the Strategy section is Purpose. Purpose encapsulates the notion either of something which a PLAN can HELP ACHIEVE or that an ORGANISATION UNIT can be responsible for. The term embraces any type of PURPOSE, regardless of whether it is on a level of organisation and time scale referred to as strategic, short term or detailed. Purpose similar to an OU can be composed or decomposed. For example a statement of Purpose may relate to something which can enable or Help-Achieve some grander Purpose. Terms like

Vision, Mission, Goal, and Objective can be denoted without the necessity for shared agreement on precisely how these terms are used. Strategy is defined as a Plan to Achieve a high-level Purpose. Strategy is based on the concept of PLAN discussed in the Activity section. The central terms to Strategic Planning can be designated with the terms Decision, Assumption, Risk, and various types of Factor.

3.7.1.4 Marketing

Sale is the key word in marketing. A Sale can be defined as an agreement for the exchange of a Product for Sale-Price that takes place between two Legal-Entities. The term Product usually refers to goods or services, while the term Sale-Price denotes monetary value, however for both terms other possibilities are included. The Roles of Vendor and Customer (usually separate) are performed by Legal-Entities. A Sale like an Activity does not specify a time period, a sale can have been agreed in the past, be taking place in the present or be envisaged as a future Potential-Sale even if the actual Product can be identified, or even exists. The Market can be defined as all Sales and Potential Sales within a scope of interest and can include Sales by Competitors. It may be decomposed into Market Segments in various ways and in many levels of detail. This can be achieved by aspects associated with the sale such as Sale-Price, Vendor, Customer and any properties of the Product. These properties are Segmentation-Variables. A Market may be analysed through understanding the Needs of Customers, the Images of Brands, features of Products, the Products, or Vendors. Promotions can be defined as Activities whose Purposes relate to the Image in a Market.

3.7.2 Definition of Terms – Enterprise Ontology

The following is a complete list of the terms defined in the Enterprise Ontology. The IDEF type description of the ontology is based on a manufacturing enterprise.

Activity	Activity Specification, Execute, Executed Activity Specification, T-Begin, T-End, Pre-Conditions, Effect, Doer, Sub-Activity, Authority, Activity Owner, Event, Plan, Sub-Plan, Planning, Process Specification, Capability, Skill, Resource, Resource Allocation, Resource Substitute.
Organisation	Person, Machine, Corporation, Partnership, Partner, Legal Entity, Organisational Unit, Manage, Delegate, Management Link, Legal Ownership, Non-Legal Ownership, Ownership, Owner, Asset, Stakeholder, Employment Contract, Share, Share Holder.
Strategy	Purpose, Hold Purpose, Intended Purpose, Strategic Purpose, Objective, vision, Mission, Goal, Help Achieve, Strategy, Strategic Planning, Strategic Action, Decision, Assumption, Critical Assumption, Non-Critical Assumption, Influence Factor, Critical Influence Factor, Non-Critical Influence Factor, Critical Success Factor, Risk.
Marketing	Sale, Potential Sale, For Sale, Sale Offer, Vendor, Actual Customer, Potential Customer, Customer, Reseller, Product, Asking Price, Sale Price, Market, Segmentation Variable, Market Segment, Market Research, Brand Image, Feature, Need, Market Need, Promotion, Competitor.

Table 3.1 Enterprise ontology terms

Having identified the key issues related to enterprise modelling as well as enterprise ontologies and discussed hoe the various methods and tools are classified, I discuss some of the most well known methods in detail and make a case for model dynamics.

3.8 Modelling Methodologies- A Static View

In the next few pages an overview of the most widely known methodologies of enterprise modelling is presented. Although we have presented an overview of the major methodologies, we're looking some of them in more detail. Generally speaking the aim of these approaches is to develop a series of schematics of an enterprise that would assist in process re-engineering or process development.

3.8.1 Enterprise modelling using CIM-OSA

[CIM-OSA 1994] is intended to model the world of manufacturing enterprises. The modeling process is concentrated at three distinct levels:

- Requirements definition,
- Design specification,
- Implementation design.

Using the modeling methodology proposed by CIM-OSA a manufacturing enterprise can create a clear view of its requirements:

'By ensuring that the physical implementation model is directly processable by the information technology (IT) components of the system, control of the operation of the CIM

systems at run time may be achieved in congruence with the specified behaviour of the enterprise.'[Tham 1995]

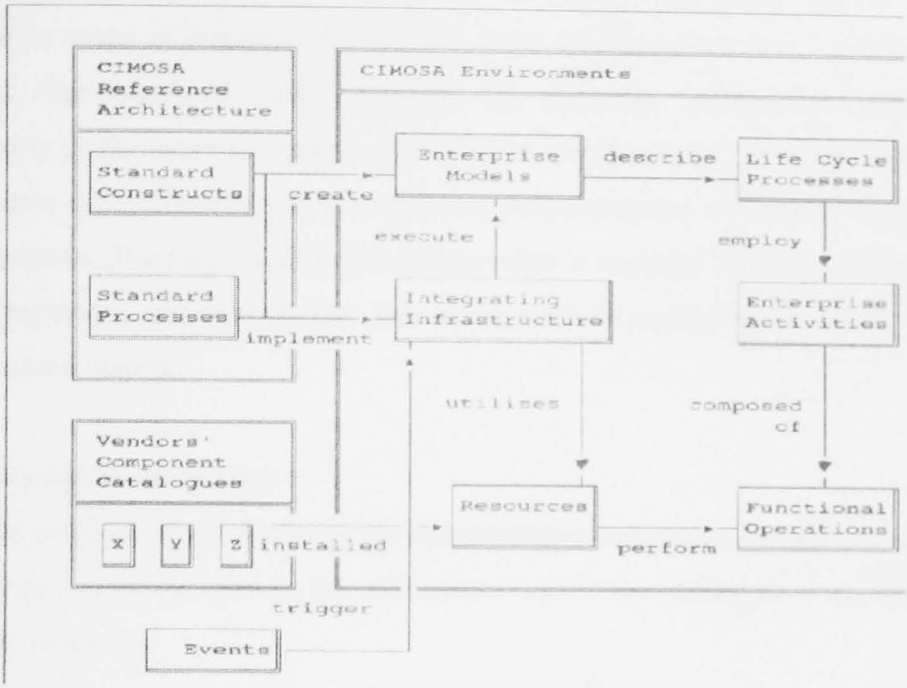

Figure 3.3:CIM-OSA Architecture

3.8.1.1 Basic Concepts of CIM-OSA

CIM-OSA does not provide a standard architecture which can be used by all manufacturing enterprises. Instead it provides a reference architecture from which particular architectures can be derived. In order to select a particular architecture *CIM-OSA* employs a number of structural concepts and architectural principles. According to [Klittich1994] the purposes they serve are:

1. to create a modelling framework of the CIM enterprise wholly or partially which distinctly segregates the WHAT (or model of required enterprise functionalities and behaviour) from the HOW (or model of an actual enterprise system implementation) by means of the HOW TO (or model of optimised enterprise system design); and

2. to derive a particular implementation model of the enterprise which is active during the operation of the enterprise system, and is the basis for the computer-controlled execution of the modelled business processes and enterprise activities, thereby providing true computer-integrated manufacturing [Jorysz 1990].

3.8.1.2 The Functional View

In order to model the functional view of an enterprise the information view has first to be modelled. This helps to establish a model of the functionality and behaviour of the enterprise in terms of domains, domain processes, enterprise activities and business processes. The functional model describes the structure, content, behaviour and functionality of the entire enterprise. *CIM-OSA* is a methodology for establishing the requirements of three levels within a manufacturing enterprise: definition, design and implementation. The functional model defines what is required in terms of structure, content, behaviour and control; how this design will be implemented as well as the actual implementation.

3.8.1.3 The Information View

CIM-OSA employs its own knowledge representation techniques in order to capture the semantics of information in the information view. The technique is divided into four major concepts:

- Generalisation,
- Aggregation,
- Particularisation or Classification and
- Generalised Relationships.

Each of these concepts is described below [Jorysz 1990].

'1. Generalisation which enables an individual object type to be thought of as a more generic object type so that type-subtype relationship definitions are established between objects or entities to propagate the IS-A hierarchy. An enterprise object can be involved in one or more IS-A links as a child of one or more higher-level enterprise objects so that this enterprise objects can inherit properties of two or more super-objects through the phenomena termed multiple property inheritance.

2. Aggregation refers to an abstraction mechanism in which an enterprise object type is regarded as a conjunctive collection of sub-component objects. aggregation defines one-to-many or many-to-many associations between enterprise objects types. This is known as the PART-OF link between enterprise objects [CIMOSA 1994].

3. Particularisation or Classification refers to the abstraction mechanism linking an enterprise object to an enterprise object type. The objects being modeled that share common properties are gathered into classes. The objects of the class are unique in the class and in CIM-OSA, this is known as the MEMBER-OF relationship between the objects and their class. Particularisation defines a one-to-one relationship between an enterprise object and its type [CIMOSA 1994].

4. Generalized Relationships refer to all other user-defined relationships between enterprise objects and are referred to as the LINKED-TO relationship in the Information View of CIM-OSA [Jorysz 1990].

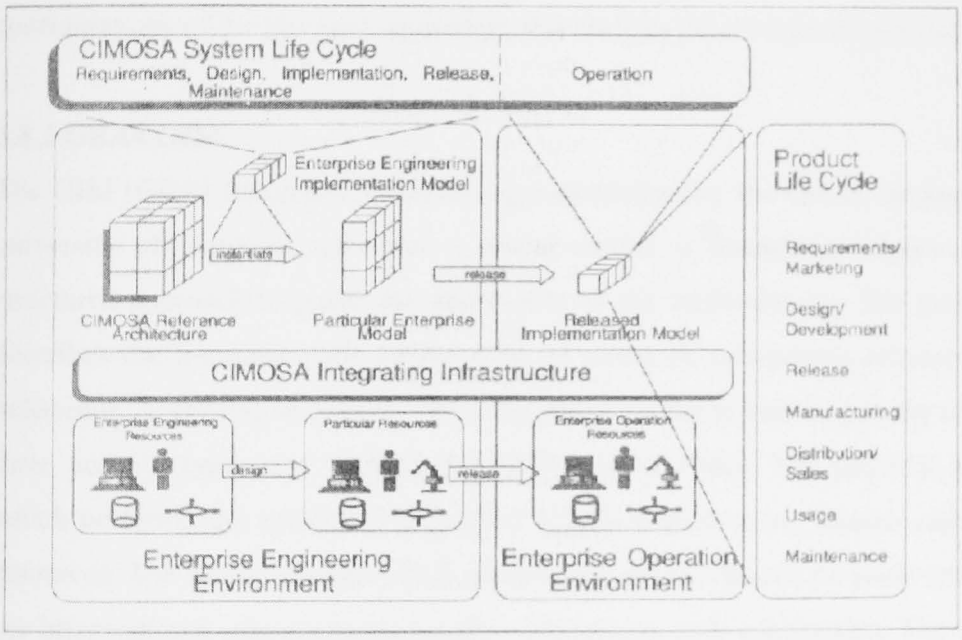

Figure 3.4: CIM-OSA Life Cycle

3.8.1.4 The Resource View

The resource view, as the name suggests, contains all the relevant information about equipment, hardware/software and people. It takes a hierarchical approach in obtaining this information, starting with the most abstract task and the resources that are needed to perform it. The task is then broken down into different levels creating a tree of tasks. Each branch has a number of people and a number of resources associated with it.

3.8.1.5 The Organisational View

The organisational view consists of all relevant information concerning responsibilities within the enterprise in terms of function, information and resources. Like the functional view it focuses on three levels. At the requirements definition level the view identifies and defines all the responsibilities that must be known within the enterprise for the human decision making process. Responsibilities may be defined for assets (resources, capital, etc.) or for operational entities, such as business processes, products and services. At the design specification level, the responsibilities for assets and operational entities have to be organised in an optimised and balanced manner in order to satisfy the decision making needs of the enterprise. Finally, at the implementation level the organisational view describes the responsibilities for configurations of the physical equipment that realises the enterprise operations.

3.8.2 GRAI/GIM

The GIM (GRAI Integrated methodology) developed by the GRAI laboratory at the university of Bordeaux provides a global model, a modelling framework and a structured approach to guide the application of the methodology. The global model describes the invariant parts of the CIM in terms of subsystems relationships and behaviour. According to [GRAI 1993] the global model is based upon the concepts of three activity types and their corresponding subsystems. The physical subsystem which performs the activities of product transformation using human and technical resources. The decisional subsystem guides production towards its goals and the third the informational subsystem feeds other subsystems with information. The modelling framework is using the IDEF methods to model the physical and functional subsystems, while GRAI formalisms are used to model the decisional subsystem. GIM's structures approach aims to cover the entire life cycle of the manufacturing system. There are four phases in the approach; initialisation, analysis, design and implementation. Initialisation is the process of describing the enterprise in terms of objectives, resources, personnel involved as well as input and output procedures. The analysis phase defines the elements of the existing system in four user oriented views. The design phase consists of the user oriented design stage and the technical oriented design stage. The user design stage is used to capture the requirements of the new system while the technical oriented stage transforms these requirements into the new system. The user requirements describe the new system in terms of information

technology and manufacturing technology. The new system is implemented during the fourth phase. GIM does not cover the operation and decomposition phases of the manufacturing life cycle. Although the approach and in particular the analysis stage may seem similar to CIMOSA in terms of views, it differs considerably in terms of context. While the decisional view is introduced the physical view unlike CIMOSA describes functional attributes and physical elements. The CIMOSA physical view however describes physical attributes of the system. The views of the analysis phase are translated into three implementation views during the technical oriented design phase. GIM's structures approach is focused on the initial phases of the system life cycle and it offers supports mainly during the analysis and design stages.

3.8.3 GERAM

With the advent of globalisation of economies, enterprises are more viewed as products themselves in the sense that they have to be designed, built and put into operation. If the enterprises already exist changes have to be specified designed and carried out [Bernus 1998]. This is a quote by Peter Bernus one of the leaders in the field of enterprise integration. During his work, he concluded that in order to design enterprises we need fundamental principles, tools and methodologies in order to support the entire life cycle. It has been observed through formal research or industrial experience that enterprises keep changing. A new business process may be automated the manufacturing procedures may be amended, the information technology infrastructure may be updated and the organisational model may be restructured.

The generic enterprises reference architecture and methodology defines (GERAM) a toolbox of concepts for analysing, designing, implementing and maintaining enterprises. The dynamic concepts that have been built into the methodology allow it to accommodate change within an enterprise environment. It is a new framework that encapsulates design concepts as well as providing an overall structure to those concepts and modelling techniques. The scope of GERAM includes the domains that need the attention of enterprise development thus the scope is defined through pragmatic need to design and redesign as well as continually improve the functioning of enterprises [Bernus 1998].

3.8.3.1 Functionality of GERAM

The functional components of a generic enterprise reference, architecture and methodology are the following:

- Generic enterprise reference architecture (GERA)

GERA is the definition of enterprise related concepts with the primary focus on the life cycle of the enterprise, since the life cycle itself is a design process GERA will have to identify the results and the components of this process.

- Generic enterprise engineering methodology (GEEM)

GEEM is a description of the processes for enterprise integration or in other words a detailed process model with instructions for each step of the integration procedure.
Generic enterprise modelling tools and languages (GEMTL)
These are the tools that carry descriptions of the models of the targeted enterprise.

- Generic enterprise models (GEM)

These models capture concepts which are common to all enterprises therefore the enterprise engineering process can use them as tested components for building any specific enterprise model.[Bernus 1998] distinguishes two levels of models; Ontological models describe the most generic aspects of enterprise related concepts. They are also considered as meta-models because the facts and rules in them are about the facts and rules of enterprise models. These types of models can capture the most basic properties of enterprise concepts such as activities, costs, dynamics etc.

Reusable enterprise models capture some common part of a class of enterprises and can be used as a building block for a complete set of models. Other reusable models describe the typical enterprise of a class while abstract models of a part of a class of enterprise capture the commonalities but leave out specific details.

- Generic enterprise modules are (GM's)

These modules are products that form implementations of components, which are likely to be used in enterprise integration, either by the enterprise integration procedure or the enterprise itself.

3.8.3.2 Advantage of GERAM

All activities that are involved either directly or indirectly in designing, operating or improving the enterprise are covered by GERAM. It provides a consistent modelling environment to support the enterprise. The modelling views offered such as the ontological view or the organisational view can be expanded to include meta models. The apparent complexity of enterprise engineering and enterprise integration has been kept low. In fact enterprise integration and enterprise engineering are both very complex tasks. GERAM treats enterprise design as a collaborative activity of multiple agents. It is also capable of tying and relating enterprise integration and enterprise engineering to the rest of the activities in the enterprise.

3.8.4 PERA

The PERA (Purdue Reference Architecture) methodology [Bernus 1996] as shown in the figure 3.4 below defines a generic information system in terms of manufacturing tasks and human based tasks. The methodology was developed in order to assist in modeling computer integrated manufacturing enterprises. Its success lies in its ability to develop an overall view of the two categorized tasks (manufacturing, human) and the interdependencies between them.

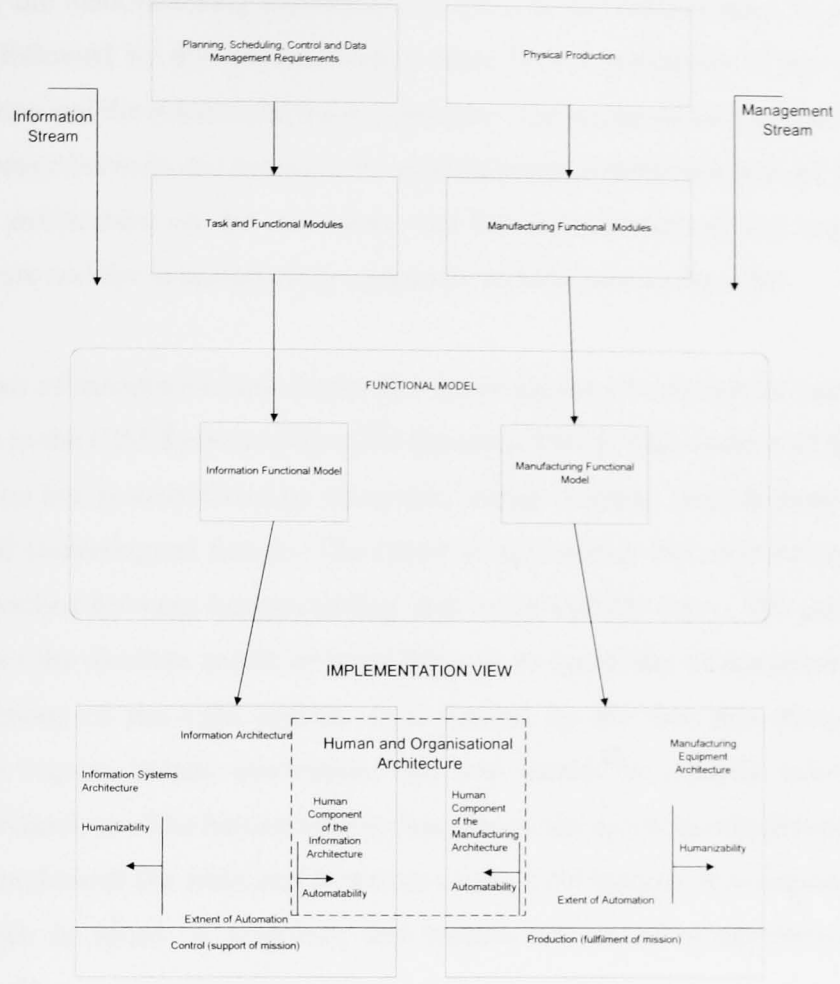

Figure 3.5: PERA Methodology

The functional descriptions of the tasks are divided into an information stream and a manufacturing stream. The information stream consists of the planning, scheduling, control and data management requirements of the enterprise. The manufacturing stream consists of the physical production requirements of the enterprise. The final model presents the enterprise from two different viewpoints: the functional view and the manufacturing (or physical) view. The functional view is composed of the information functional model and the manufacturing functional model. The functional view is followed by the implementation view which is composed of the information architecture and the manufacturing (or customer service) architecture.

The information functional model of the functional view primarily provides the input for the formation of the information architecture of the implementation view, whereas

the manufacturing functional model of the functional view primarily serves as the input for the manufacturing architecture of the implementation view. The functional view is followed by the implementation view that is composed of the information architecture and the manufacturing architecture. The extent of automation line defines the boundary between the human and organizational architecture and the information systems architecture on the one hand, and between the human and organizational architecture and the manufacturing equipment architecture on the other.

The extent of automation line shows the actual degree of automation carried out (or planned) in the CIM System of the CIM Business Entity. The location of the extent of automation line is determined by economic, social, customs, laws & directives, union rules, and technological factors. The extent of automation line may be considered to be sandwiched between humanizability and automatability lines. The automatability line shows the absolute extent of technology in its capability of automating the tasks and functions of the CIM system. It is limited by the fact that many tasks and functions require human innovation, etc. and cannot be entirely automated with current technology. The humanizability line shows the extent to which humans can be used to implement the tasks and functions of the CIM system. It is limited by human capabilities in speed of response, and human powers of comprehension, vision, strength, etc.

The information architecture of the implemented view lends control features like human decision making, and human monitoring of information systems. On the other hand, the manufacturing architecture of the implemented view provides production facilities for the fulfillment of the enterprise's mission.

3.8.5 The enterprise project

The Enterprise project promotes the use of knowledge-based systems in enterprise modelling and aims to support organisations effectively in the Management of Change and is the UK government's major initiative with a budget of £2.6 million. The focus of the project is to help manage change through the strategic use of IT and management innovation. The project supports enterprise models that encapsulate how a business works and how it is organised. The major objective of enterprise modelling is to gain an enterprise-wide view of an organisation and which subsequently can be

used as a platform for making decisions. The Enterprise project developed an Enterprise toolset, which uses executable process models to enable users to perform tasks. The toolset is executed via an agent-based architecture to integrate off-the-shelf tools in a plug-and-play style. The approach adopted by the Enterprise project is utilised to address the major problems such as the impact of change, communication problems, responsiveness, process consistency and IT systems.

3.8.5.1 Business Perspective

It is essential for businesses to increase both their relative and absolute performance due to a combination of factors both internal and external to the business. Examples of such factors can be listed as the need to improve financial performance; customer satisfaction; adapt to periods of growth and recession and decrease cycle times. To ensure that management of change is successful it is important that businesses monitor and improve their performance against strategic objectives. This ability should be supported by methods and tools which enable modelling and which analyse and improve various aspects of how a business works and is organised. This process requires that existing modelling methods are improved and if necessary replaced by a framework which integrates methods and tools appropriate to enterprise modelling and the management of change. Providing a method and computer toolset to capture aspects of a business and analyse these, in order to identify and compare options for meeting the business requirements, is the aim of the Enterprise project.

3.8.5.2. Technical Perspective

An Ontology for enterprise modelling is the foundation upon which the framework for integrating tools and methods is built. This framework supports a general base of practical knowledge based modelling tools and methods for business application and evolved alongside existing and emerging standards in open systems and knowledge representation.

3.8.5.3 The Toolset

The Enterprise Toolset utilises the integration framework. It initiates an agent-based architecture to integrate off-the-shelf tools in a plug-and-play style. The Enterprise Toolset is a composite, these components include a procedure builder for the apprehension of process models, an agent toolkit for supporting the development of

agents, a task manager for integration, visualisation, and support for process enactment, and an enterprise ontology for communication. The Procedure Builder is a graphical tool for explaining and recording business process models. The output from the Procedure Builder can be exported for use directly by the Task Manager. In addition the Procedure Builder can produce reports containing the process diagrams and associated process information. The Procedure Builder permits users to graphically capture their business procedures in a form which can be used by the Enterprise Task Manager. Procedure diagrams contain procedure nodes, linked together by precedence arcs. The diagram can also include conditional fan-in and fan-out branches. Procedures can be hierarchical in nature and are composite with procedures at one level being capable of being broken down into further procedures. Procedure Builder utilises part of the IDEF3 Process Description Capture Method. In Particular, it uses the notation for the Process Flow Networks (PFN's) that are one of the two different views on processes provided by the IDEF3 method. The procedure builder allows the user to do a variety of tasks users are able to build diagrams that are true IDEF3 PFN's and users can fill in elaboration forms for procedures, junctions and links, using forms that are identical to IDEF3 elaboration forms. The Procedure Builder allows the user to print out individual procedure diagrams, or all the diagrams that make up the procedure. The user also has the option of printing out a report on the procedure that contains both diagrams and the information that is contained in the elaboration documents for the diagram. The main function of the Procedure Builder for the Enterprise project is to provide procedures as input to the Task Manager in the Enterprise tool set. The goal of printed reports and diagrams, while useful to the user, is to produce output that is useful for the Task Manager. It is for that purpose that the Procedure Builder allows the user to add more information to a procedure diagram that will be used by the Task Manager, examples of such information are precondition and effect information for procedures, and marking start and end nodes for procedures.

For the agent-based architecture of the Enterprise Toolset we investigated a multitude of externally available solutions and concluded, at that time, that none of them were suitable and thus developed our own agent-based solution supported by the Agent Toolkit. The design of the toolkit was guided by one principle; to make the creation of new agents as unproblematic as possible. It was essential that the toolkit should be

able to accommodate as-yet unknown tools as agents without the need to redesign the Agent Toolkit or any other component of the Enterprise Toolset. It was also essential to ensure that an organisation's existing applications and tools (i.e. their legacy systems) could be utilised. The communication facilities for applications in the Enterprise toolset are provided by the agent toolkit. The agent toolkit is used to turn applications into Enterprise agents ("agentification"). Those who will be turning user applications into Enterprise agents require knowledge about the agent toolkit, while 'End' users do not.

The interface between the user and the Enterprise Toolset is known as the Task Manager and it is used directly to support the user in performing their current tasks. Planning user tasks and the utilisation of agents, based on the information available from the Procedure Builder's process models and agent registration information, is the job of the Task Manager. Suitable agents are identified at the last possible moment ensuring that the most suitable agent at that time can be identified. The Task Manager is also responsible for monitoring the progress of a task, keeping note of which tasks are currently active and which have been completed, etc. This progress is visualised and supported by the process diagrams captured with the Procedure Builder. The Task Manager also helps the user to recover from failures by determining alternative routes of action. Essentially the Task Manager provides extra control on top of those provided by the agent services. The user is able to participate in the co-ordination of the use of agents at the level of the user's tasks depending upon tasks the user is engaged.

The Enterprise Ontology was developed as standard terminology for use in the Enterprise Toolset. This overcame problems experienced by tools that were developed separately, such as conflicts and ambiguity when tools are integrated. The Enterprise Ontology is a set of generally applicable terms used frequently in enterprises, each term is carefully defined to allow common usage. Each individual organisation will have their own set of terms thus the ontology can be extended to suit the specific needs of the organisation. The utilisation of this ontology is advantageous due to the fact that terms are used consistently and unambiguously throughout the enterprise. The ontology achieves the basis for communication between agents regardless whether they are human or software agents.

3.8.5.4 Applications

The evaluation of the toolset in the context of real business applications was enabled through the addition of end-user organisations such as Lloyd's Register, Unilever and IBM . Each end-user organisations had different uses for the evaluation of the toolset. Lloyd's Register uses the results for strategic planning through more effective modelling and re-engineering of business processes. Unilever first uses the toolset within its R&D activities while IBM UK intends to exploit the results in remodelling its internal organisation as well as providing technical input via its Business Modelling Method BSDM (Business Systems Development Method).

How pragmatic methods and current commercial tools can be used to support business process re-engineering were addressed in the early stages of the project by Logica. The main developer of the toolset (AIAI), introduces a world-leading technical capability in KBS technology and applications to the consortium, this together with Pilkington Optronics resulted in a public demonstrator which addresses the problem of bid management. The benefits of the project are represented to the wider business community by the partners themselves. The key public deliverables are:

- a review of enterprise modelling techniques, tools and methods;
- two early demonstrators using off-the-shelf software;
- the public demonstrator of the toolset applied to the generic business problem of bid management;
- the Enterprise Ontology;

3.9 Enterprise Modelling Methodologies – A Dynamic View

The following paragraphs present a number of modelling methodologies that exhibit dynamic behaviour. The most well known of these methods is the IDEF models. The chapter also reports on WADE as well as TOVE. TOVE is different from the methodologies examined so far in the sense that it targets an aspect of the enterprise which had not received much attention, that is enterprise communication. The models that TOVE creates are based on computation representations of the enterprise rather than diagrammatic.

3.9.1 IDEF methods

The integrated definition method (IDEF) is a structured approach used for enterprise re-engineering and BPR [DeWitte 1998]. It uses visual models that facilitate the quantitative analysis of proposed changes to processes to yield the highest performance at he lower cost. Generally speaking modelling assists enterprises to understand their processes in terms of how they work, what input they require and what output they produce. By developing models and understanding of the 'as-is' of the enterprise one can visualise a 'to-be' situation and assess it and implement. This is where the value of the IDEF method lies. It offers support into developing 'as-is' and 'to-be' models or in other assists in describing where a company is and it's going. There are numerous IDEF methods such as IDEF0, IDEF top level or IDEF3. Each of these methods targets different aspects or levels of the enterprise. IDEF0 defines the essential elements of each process and describes the relationships between processes. IDEF top level emphasises on functional modelling while IDEF3 provides a process description that serves as the basis for simulation models. In this chapter I have concentrated on the IDEF methods as a whole rather than a specific one. The following figure shows IDEF's basic syntax.

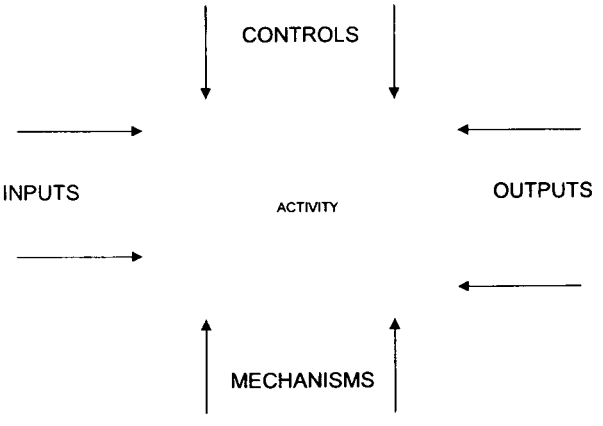

Figure 3.6: Basic IDEF Syntax

IDEF methods are used in turn according to the stage of analysis. IDEF0 initially is used to capture, visualise what the enterprise does in terms of functionality. It creates models of decisions, actions, activities, inputs, outputs and constraints. The underlined models are developed by capturing the elements needed to execute a process, models the relationships between each process, identifies core processes as well as non value-added processes and finally assists in activity based costing scenarios.

According to [DeWitte 1998] the most powerful feature of IDEF is its lack of reliance on time. This non-linear approach frees management from preconceptions of how the process occurs by modelling how activities actually relate to one another. By removing these preconceptions IDEF provides an objective assessment of what really occurs in the process. The accurate 'as-is' picture forms the basis for future improvement.

IDEF3 is a process description method. It describes how processes flow and how activities are linked together in order to form a process. In essence, IDEF3 provides the real world picture of the enterprise or in other words the what it does and how it does it by capturing also the timing and decision logic involved with each process. IDEF3 has become a popular way to model for the modelling of 'what if' scenarios prior to discrete event simulation. A high level process is made up of individual activities linked together. IDEF3 is capable of decomposing high level processes into their smallest activities or units in order to deal with complexity. Like CIMOSA, PERA and GIM the IDEF methods allow modelling from multiple perspectives. The various perspectives that relate to a particular process once captured and visualised can be used to achieve an even closer interpretation of how the process actually works. The element that makes IDEF distinguishable from other modelling methodologies is that it accounts for random behaviour and its effects on a process. IDEF can also develop object oriented models through object state transition network descriptions. IDEF3 can capture objects associated with a process in a similar way IDEF0 captures resources. IDEF methods can also be linked. IDEF3 takes the knowledge captured by IDEF0 supplements it with a process owner's real world knowledge and how a process works, and produces robust process scenarios that can feed directly into simulation scenarios [DeWitte 1998]. The basic construction of the IDEF model is shown in the following figure.

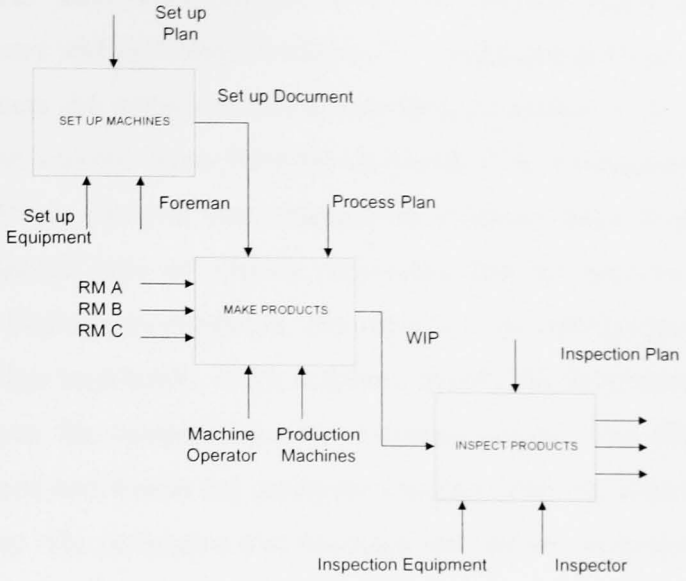

Figure 3.7: Basic Construction of an IDEF model

The IDEF5 method has three main components that are; graphical language, a structured text language and semantic procedure. IDEF5 provides several schemas for the visualisation of enterprises. Classification semantics are used in IDEF5 to show relations between kinds in IDEF5 model. It supports two types of classification mechanisms which are; description subsumption and AKO "a kind of" as shown in the following figure. In description subsumption, the defining properties of the "top-level" kind K in the classification, as well as those of all its sub-kinds, constitute rigorous necessary and sufficient conditions for membership in those kinds. Additionally, the defining properties of all the subkinds are "subsumed" by the defining properties of K in the sense that the defining properties of each kind entail the defining properties of K; the defining properties of K constitute a more general concept.

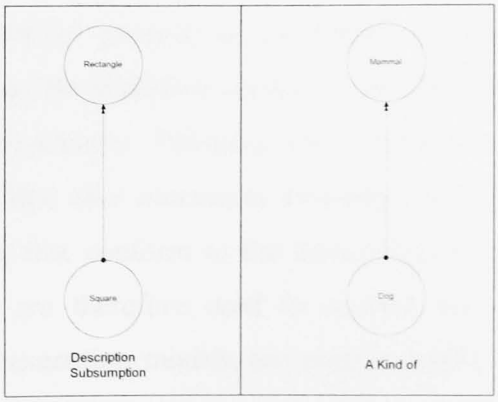

Figure 3.8: Classifications Mechanisms

Conversely, natural kind classification does not assume there are rigorously identifiable necessary and sufficient conditions for membership in the top-level kind K. Nonetheless, there are some underlying structural properties of its instances that, when specialized in various ways, yield the sub-kinds of K. Composition schematics are used in IDEF5 to discover and characterise different uses of the part-whole relation. It is a special type of relation schematics that are used to represent the relations between kinds in an enterprise. The capture of knowledge about relations is critical to knowledge acquisition, since relations specify the behaviour that controls interactions between the components of a complex system. The IDEF5 language provides a structured text format for capturing complex relation of knowledge at any level of complexity. The structured text language can express everything that can be stored using the schematics.

The IDEF5 procedure consists of five activities. The first activity establishes the purpose, viewpoint and context of the enterprise. The second activity is concerned with data collection. It acquires the data needed for the development of the enterprise model, while the third and fourth activities are used to analyse data and develop the model respectively. The fifth activity is concerned with validating and completing the enterprise model development process.

3.9.2 WADE method

WADE (workflow analysis and design environment) is a modelling method dedicated to the support of design and analysis of workflow systems in the context of continually evolving business processes. It supports concurrent design of the business process with the workflow process that must support the business process. In today's environment a business process is developed by process owners or business stakeholders, whereas the workflow process is developed by the information systems engineers and programmers. Bringing concurrency to these activities results in business processes that take maximum advantage of the workflow automation and workflow processes that conform to the business goals [Parakath 1998]. Workflow simulation models are therefore used to analyse and improve the process flow whereas workflow execution models are used to implement the process within the context of an information system.

The WADE model consists of 8 phases. The first phase is concerned with acquiring business process descriptions. At this phase the knowledge of the business processes is acquired from domain experts and stored in a knowledge representation format. WADE facilitates the acquisition of this information using the IDEF3 process description capture model. The second phase is concerned with the design of a business process simulation model starting from a business process description. WADE supports this phase by making use of its BP model designer component to develop these simulation model design tasks. The process id based on pre-determined rules which have similarities with the dynamic model presented in the next chapter. For example:

If the inventory level is greater than the requested material quantity
Then perform transfer material
Else perform generate purchase order

In the third phase WADE executes the model of business process simulation which is executed under a set pre-specified experimental conditions. The executions yielded output data that is recorded and compiled. These tasks are supported by the simulation engine component of the WADE architecture [Parakath 1998]. WADE also includes functions for accessing and editing model information, viewing key performance metrics while simulation executes, and finally observing the animation of the model. In the fourth phases WADE acquires the description of the workflow. The difference between workflow description and business descriptions is that the former concentrates on the flow of single information work-items that triggers or controls an instance of a business process. In complex systems work-items any tie together multiple business processes. WADE acquires this type of knowledge again by using the IDEF3 process description method. The description themselves contain information about the applications needed within the enterprise as well as the information needed to launch them. One of the features of WADE that distinguishes it from other modelling methodologies is that it can generate workflow models from the workflow description. The fifth phase of the approach deals with the design of the workflow simulation models. These are representations of processes that can be directly executed by a discrete-event simulation engine [Parakath 1998]. In the sixth phase of the approach the workflow simulations are executed for experimental

purposes. The data the experiments provided are later recorded and compiled. The analysis of the recorded data is performed in order to identify potential improvements to the workflow description. The seventh phase comprises the workflow execution model which is a representation of a workflow process that can be directly executed by the workflow engine. The description of the workflow execution model is again based on IDEF3. The final phase is the execution of the workflow itself. The workflow specification models are used to generate the workflow system which is then used to automate the flow of work. The WADE conceptual model is shown in the following figure.

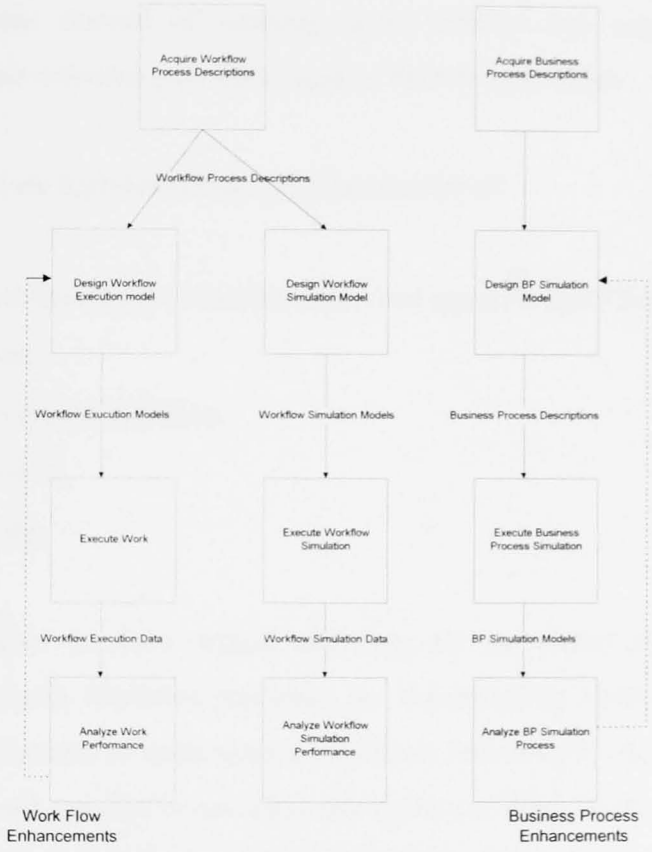

Figure 3.9: WADE Concept of Operation

The developers of the WADE model claim that the model facilitates the process of planning within enterprises as the WADE models can be used for analysis and planning of tasks. It is also capable of identifying opportunities for BPR and evaluating alternative business process designs. Finally it allows process modelling and simulation to be used in parallel to design and engineer workflow systems.

3.9.3 Enterprise modelling with FUNSOFT nets

FUNSOFT net describes complex processes for distributing development large scale systems using large scale tools such as CORMAN, or LEU [Erdmann 1997]. FUNSOFT nets are structured and formalised descriptions of work flow models. The developers [Erdmann 1997] called them high-level Petri nets, which offer different views on a model (process view and project management view). Flexible and adaptable firing behaviour tailored for business applications, as well as formally defined workflow models in terms of predicts-position nets. Petri nets enforce a clear separation of data and computation. This separation is desirable on the level of business process, as it is usually easier for domain experts to describe how things have to be done instead of thinking about clashes and responsibilities which contradicts object-oriented paradigms against Petri net paradigm.

FUNSOFT nets are high-level Petri nets that consist of:

- A petri net structure (s,t,f) (s for channels, t for agencies and f for edges),
- A set of services,
- A set of object type definitions,
- A set of predicates,
- A initial marking

Channels are used to store objects and they are associated with an object type (Boolean, real, text). Business processes are described by agencies. The input and output firing behaviour of each agency is defined individually. Agents can be refined in terms of the sub-net that is run when the agency is fired. Each non-refined agency has an associated service that is invoked when the agency fires as well as an agency role that describes the kind of persons involved in the activity. Edges between channels and edges are divided between standard Petri net consuming edges and copying edges which provide read access to tokens without removing them from their channel. In order to deal with complexity FUNSOFT defines different views. The basic views are the process view that describes the structure of business processes, the project management view that indicates the relationships between roles and activities and the object-type view that defines the information structure used in the model.

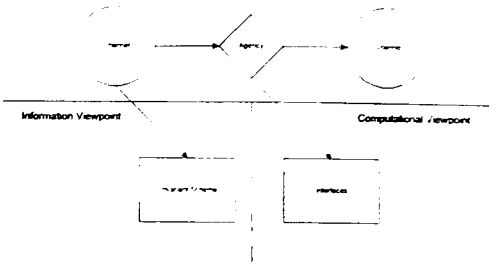

Figure 3.10: Channels and Agencies

3.9.3.1 Information & Computational Viewpoint

FUNSOFT nets are used to specify a business objective including actors and all necessary activities. The channels that were described earlier are used to capture the resources needed for a business process. Agencies represent individual activities within a modelled community. Often agencies operate on tokens from several channels simultaneously, in such cases the relation between the agency and an interface is not that clear. Mapping the agency for generic function that will be performed by all objects, rarely suits real life situations. FUNSOFT net models a business process as a series of communicating objects. All agencies in this community (model) have to be related to FUNSOFT roles. This concept controls the firing of agencies in the sense that they involved FUNSOFT roles have all the necessary access rights to invoke the functionality of the agency [Erdmann 1997]. Agencies are annotated using user-defined attributes that can also be used to describe the quality of service for individual agencies in the enterprise viewpoint. This data is mapped to interfaces in the computational viewpoint and can be used to evaluate the applicability of the implemented system.

The developers of this approach claim that enterprise modelling using FUNSOFT nets is a promising approach for the specification and integration of large scale distributed heterogeneous information systems. Focussing on business processes suits the current organisational trends in industry, summarised through the term business process and engineering. The approach encourages the participation of system-users which leads to higher acceptance of future systems and defines a basis for reuse later on. The decomposition into enterprise objects will be realised as self-contained entities focusing on specialised functions. When a particular business process needs to be changed, the objects (that describe the process) are just being reconfigured. Another advantage of the approach is to simulate new business processes. New business

processes can be simulated by reconfiguring existing enterprise objects that are evaluated by assessing the impact of change upon them. Simulating new business processes will increase the acceptance of the future system. Since user participation and collaboration with the analysts is strongly encouraged, emphasis can be given to integration between old and new processes. Even in this process the end-users will not lose track of the systems development since possible problems affecting the inputs and outputs of business processes can be resolved between them and the systems engineers.

3.9.4 The TOVE Methodology

TOVE (Toronto Virtual Enterprise) [Fox et al 1996] is an ontological approach which aims to create a common and shared terminology in an enterprise. TOVE defines the meaning of each term (semantics) in an easy to understand, and describes these semantics as a set of axioms. TOVE finally defines a set of symbols for depicting a term in a graphical form. According to [Fox 1996] all of the attempts to create a general enterprise model fail to produce a set of criteria against which knowledge representations can be evaluated. Some of the criteria suggested in his paper are:

- Generality,
- Competence,
- Efficiency,
- Perspicuity,
- Transformability,
- Extensibility,
- Granularity,
- Scalability.

The first two criteria examine to what degree the representation is shared and how well problem solving is supported. The third and fourth criteria examine whether the representation supports efficient reasoning and whether it is easily understood by users. The transformability and extensibility criteria ensure that the representation can be transformed into another, more appropriate representation for a particular domain, and can also be extended to encompass new concepts. Granularity ensures that the

representation supports reasoning at every level of abstraction and detail, and finally scalability examines whether the representation can be scaled up to support large applications.

The philosophy behind TOVE is that enterprises are action oriented and therefore the ability to represent actions lies at the heart of all enterprise models. TOVE developers approached the issues raised by the evaluation criteria by defining a generic level representation of activities, time, causality and constraints. In TOVE, actions are represented by the combination of an activity and its corresponding state. Activities are the basic transformational action primitive via which processes and operations can be represented. *Enabling state* defines what has to be true prior to the implementation of an activity, and *caused state* defines what has to be true after an activity had been performed. An activity may at any point carry a status, which may be either

- Dormant: the activity is idle,
- Enabled: the activity is executing
- Completed: the activity is finished or
- Suspended: the activity has been forced to idle state.

The *state* of an activity is another term and indicates what has to be true for an activity to be performed. The above form the terminology of the *TOVE model.* The definition of the terminology is in the form of first order logic and has been implemented in Prolog. An English description [Fox 1996] of some of these definitions are:

- An activity can be executed if its enabling state is enabled.
- A resource is physically divisible if it has at least one sub-component.
- A resource is reusable if it is temporarily divisible in its role in an activity.

3.10 Dynamic modelling methods and tools

In the following paragraphs some dynamic models are being presented. Some of them target supply chains while others target specific processes within an enterprise. Supply chains are presented as networks of processes expressed using diagrammatic notations while processes are expressed as mathematical models.

3.10.1 A multi-agent approach to modelling supply chain dynamics

The following paragraphs introduce the research work of Swaminathan, Smith and Sadeh [Swaminathan 1997] into the area of supply chain dynamics. The approach utilises a multi-agent paradigm for modelling and analysing supply chains. The aim of the approach is to achieve supply chain optimisation by experimenting with different scenarios. Supply chains are considered multi-agent environments as many different business entities are working closely together and are therefore interdependent upon each other. Each agent in the supply chain has the ability to utilise various policies relating to demand, supply, information and materials control. The analysis is based on discrete event simulation of the various alternatives and control policies. All elements of a supply chain are classified as structural or control elements. Structural elements are involved in actual production and transportation of products whereas control elements help in co-ordinating the flow of products in an efficient manner with the use of messages.

Every enterprise in the supply chain is regarded as a specialised agent. A manufacturing agent for example is different from the distribution agent or a transportation agent in terms of objectives. Therefore specialised elements correspond to the structural elements of the supply chain. Agents communicate with each other by passing around messages. The framework has developed a messaging mechanism for identifying the agent each message activates.

The idea behind the framework is based upon the development of a knowledge based system that holds a set of predefined rules. These rules handle the type of relationships and messages that can exist within the scope of the supply chain. In the following paragraph we describe how each off the agents involved in the model expected to behave.

As we mentioned in the beginning the approach divides the supply chain entities between structural and control elements. The business entities come under the heading of structural elements, whereas policies regarding the flow of material as well as the way the means by which the material flows is described by the control elements. Production elements use inventory control elements for managing their stock, flow

control elements for transportation and forecast elements for propagating demand forecasts. The authors have recognised five agents within the scope of the supply chain. The behaviour of each one of those is defined in a knowledge base whichi forms the ontology of the model. These agent definitions are in a sense generic objectives of business entities. A retailer, for example, would focus on reducing the time from the delivery of a customer order and simultaneously minimise stock. Furthermore, a distribution centre agent that receives goods from the manufacturer would focus on reducing the stock carried. Note here that the distribution centre agent is either storing goods or sending them away to retailers. Manufacturing agents assemble or develop goods from scratch. Generally speaking, they would accept orders from either distribution agents or retailer agents. Their focus would be on stock control but mainly on the manufacturing process. The transportation agents-as the name suggest-are responsible for moving products from one agent to another. The focus of this agent would be on managing delivery times and general logistics. Finally, the authors consider a supply agent whose aim is on low turn-around time and inventory. There is a number of issues raised by this approach. There is a number of unique processes each working towards a specific goal. Although enterprise modelling can offer a modelling approach for each one the goals of the entire objective would not have been met. The issue here is dependability. The functions presented here are interconnected and dependent upon each other. The output of a process affects the output of the others. It has been concluded during this research that it is environments with multiple goals that need dynamic models to assess them. We need therefore a set of initial rules that capture the behaviour of the entire chain in order to assess the affect of each process to the entire model. In the following paragraph we go into detail in the approach developed during this research for developing dynamic models and assessing the impact of each component to the entire picture.

The control elements are policies that control production and transportation of products within the chain. The inventory control element controls the flow of materials and can either be centralised or decentralised. The centralised parameter of the inventory control would take into account the entire chain whereas the decentralised parameter focuses on one specific entity. The demand control element denotes actual forecasts and are modelled as messages within the scope of this

framework. Again demand control elements can either be of a marketing or a forecasting nature. The marketing element investigates ways to increase demand for a particular product and could be in the form of advertising, discounts or seasonal sale. From a programming point of view the parameter provides a mechanism that triggers additional sales. Forecasting demand elements determine how forecasts are generated and evolve within the supply chain. The supply control element carry the terms and conditions of the goods. The terms and conditions which are part of contractual arrangements specify variables such as price, delivery times etc. The floor control elements co-ordinate the product flow and are divided into loading and routing elements. Whereas loading elements control the manner in which the transportation elements are loaded and unloaded, the routing elements controls the sequence products are delivered by the transportation elements. Finally the information control elements is divided into directly accessible information such as information on stock level, capacity information or routing and periodic information which refers to changes in business strategy and pricing introduction of new services or products. The diagram below describes the relationship between structure and control elements.

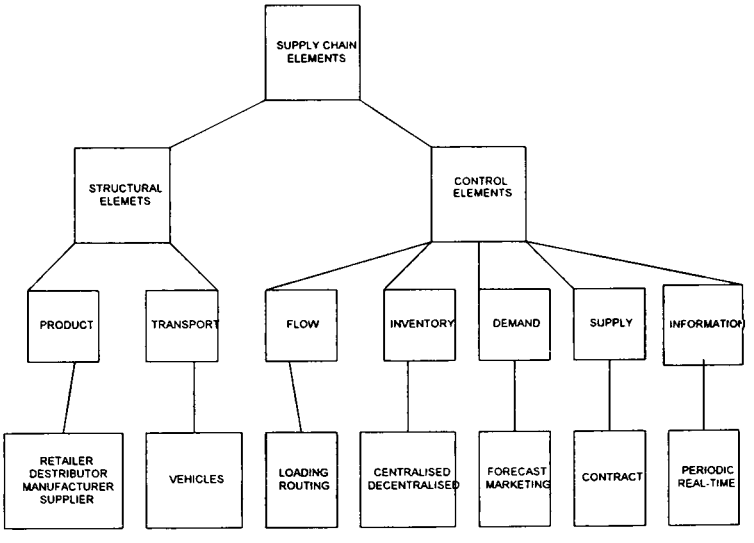

Figure 3.11: Multi-agent approach to supply chain modelling

The framework is aiming to assist analysts, managers or developers during supply chain re-engineering stage. The framework provides a platform for testing different scenarios rather than undertaking a probabilistic view of the future and carry out risk analysis. The authors claim that the framework can be used as a decision support tool that can analyse various alternatives and be useful in quantifying gains and helping the organisations make the right decision.

The following example shows how the model reacts to a raised order. It shows how structural and control elements interact. Assume an automotive supply chain and consider the following example [Swaminathan 1997].

- *An enterprise has initiated a request for goods message*

The message handler performs the following actions following receipt of the message.

Checks if the product is available in stock. If it is then demand is satisfied and inventory is updated, otherwise demand is backlogged and the status of backlogged demand is updated.
The inventory policy control is invoked.
The inventory policy control generates a request for goods message for the supplier of the product based on inventory on-hand and backlogged demand. It also utilises supplier capacity information based on agreements for information sharing with the supplier.
If outgoing messages are generated they are queued up in the message queue with the time stamp for activation.

3.10.2 A dynamic transportation scheduling model

The following paragraph presents a model developed as part of a Ph.D. thesis by [Scott 1995]. The model, as the title suggests, attempts to model a transportation chain for the purpose of scheduling optimisation. The author distinguishes the methods for craft scheduling into direct and sequential methods. Direct methods cover single step or integrated models whereas sequential methods cover multi-step models with minor feedback. It should be stressed here that this particular model targeted aircraft scheduling. While some successes with direct are reported nearly all craft scheduling techniques are sequential. [Scott 1995] believes that there are mainly two reasons for this. Most experienced craft schedulers use sequential techniques, and the sequential method computation time grow less quickly with problems size than do direct methods. However sequential methods sometime rule out attractive alternatives in the early steps of computation, in order to make the method more tractable. His model

explored these deficiencies and used a decision vector model for craft scheduling in conjunction with the Lagrangian relaxation algorithm. The model identifies the optimal schedule. The sequential model is following the familiar path of building a set of candidate schedules which are later evaluated and the optimal one is chosen. Schedules are evaluated against three parameters which are combined cargo (includes passengers) craft only cargo and craft movement constraints. The author also claims that this modelling technique maintains a global optimality by using Lagrangian relaxation to produce a set of craft schedules containing the optimum craft and cargo schedule.

3.10.3 NIMBUS

The NIMBUS model is a software package designed for dynamic modelling and simulation of process plants. The developers however claim that it can simulate any system that can be described by differential and algebraic equations. By separating the tasks of modelling and simulation, and providing a suite of DAE (Differential Algebraic Equation) solvers, the NIMBUS user can concentrate on the modelling task [Nimbus 1998]. Process models are configured in the form of a flow sheet, using an object-based graphical configurator. Objects are chosen from libraries of sub-models. Users can also create their own libraries and sub-models making the tool very flexible.

NIMBUS was developed at the University of Queensland, by the Chemical Engineering department. It has modelled a number of lumped and distributed parameter systems. NIMBUS contains a set of computer programs (functions) that facilitate the modelling and simulation of processes that can be represented in terms of semi-implicit differential and algebraic equations of the form:

$$dx/dt = f(x, z, p, t)$$
$$0 = g(x, z, p, t)$$

with states x, algebraic variables z, and parameters p. The independent variable is usually time t.

Numerical techniques are used to solve the dynamic and steady state (dx/dt = 0) problems. These include DIRK (diagonally implicit Runge Kutta) adaptive step-length integrators and a Schubert algebraic solver for initialization or steady state solutions [Nimbus 1998].

To the user, process systems are graphically configured in terms of objects called blocks and links in the form of a flow diagram. Block objects are selected from libraries of pre-programmed sub-models and connected by pre-defined link objects. However, the assembled model is viewed numerically as one large DAE (Differential Algebraic Equation) system, and so-called equation-based solvers are used to perform the simulation. A variety of programs are included to build libraries, to view results in graphical and tabular form, to diagnose modelling problems, and to link to other software such as spreadsheets and MATLAB (for system analysis and control system design). Versions of the dynamic solver have also been linked to model-based controllers, to the G2 real-time expert system , and to the FSQP optimisation package.

NIMBUS aims to facilitate optimal processing. It can contribute to improve the design of the process and its control system, to optimal policies for start-up and other disturbance handling, to optimise operating conditions and control system tuning, to "what if" exploratory investigations and to operator training.

Dynamic simulation can detect sensitivities of the plant to particular types of disturbances which small changes in the design can reduce. Simulation can also highlight start-up and shut-down problems. NIMBUS and associated MATLAB tools can be used to examine interactions between variables that may otherwise be unsuspected.

NIMBUS is a tool for designing and evaluating control system designs. The NIMBUS MATLAB toolbox contains an IMC tuner for PI control loops. NIMBUS and other MATLAB toolboxes can be used for model reduction, estimator design and the design of advanced optimal controllers [Nimbus 1998].

It is generally desirable to simulate the dynamic response of a plant to strategies for start-up, product grade or feedstock changes, and schedules for batch processing.

Optimisers which have been integrated with NIMBUS can also assist in developing these operating strategies. "What if" scenarios are one of the traditional uses dynamic simulators. What is the effect of different feed stocks, operating conditions, throughputs, plant or control system designs, additions or removal of plant equipment, to name just a few. Another traditional use of dynamic simulators such as NIMBUS is in operator training. Operators can practise and develop strategies for handling start-up, shut-down and emergencies. This is seldom possible and generally much more expensive on the real plant. The tools can run on personal computers or laptops It comes as an OS/2 version and the hard-working parts can also run on UNIX workstations.

The developers also claim that the numerical engines and strategies within NIMBUS are the very latest technology and will continue to be so through ongoing research. NIMBUS itself and the NIMBUS solvers are easily integrated with other software such as display and interfacing software, control systems, expert systems, design and analysis tools. It has already been interfaced to MATLAB, G2 (a real time expert system), spreadsheets and databases.

But perhaps the key difference between NIMBUS and other simulators is its transparency and open architecture. This has been achieved by a modular structure, an object-oriented design, no binary data files, all model equations and parameters accessible, and operating log files. This open architecture makes it particularly simple to define new link types, write new blocks, incorporate FORTRAN code for all or parts of blocks, interface properties packages, and integrate NIMBUS with other software tools.

3.10.4 A decision support model for modelling logistic chains

The following paragraph examines the SMILE (Strategic Model for integrated logistic evaluations) project [Tavasszy 1997]. The model was constructed as a joint effort of the transport research centre of the ministry of transport and the research organisation NEI (Netherlands Economic Institute). SMILE is a decision support system which describes logistic chains at three levels: production, inventory and transportation. The developers of SMILE had realised prior to the development of their model that private and public decision makers are becoming more dependant on strategic information on

expected future development of freight flows. Bearing in mind the dynamics of global logistic processes, the required strategic information is no longer limited to forecasts of the use of the transport system in a specific year in the future. In order to assist strategic management there is a need for insight in trends throughout the years. It should be stressed at this point that decision making is a three stage process. The process is divided into the stage of perception (information gathering), combination (comparison of available options) and decision. SMILE has recognised the relation that exists between transport and the economy and the impact it may have. [Tavasszy 1997] claims that the model can make forecasts based on 'what if' scenarios that involve certain economic circumstances. A typical use SMILE is to assess for example what is the influence of central European distribution on transport, or what is the contribution of transport to the economy. There are of course a number of parameters that need to be quantified in order to reply to such questions. By taking into consideration variables such as modes of transportation, economics and the dynamics between the two SMILE has the ability to analyse logistical choices and make long term forecasts. The SMILE model aims to get a better view of future developments in freight flows that use Dutch infrastructure.

3.11 Commercial Products

In the following paragraphs we're looking into some of the commercial products available on the market. Since we distinguished between static and dynamic research methodologies and tools we're using the same distinction for this section too. The commercial products have also been divided according to the classification system used to divide between static and dynamic systems. During the research into the commercial products available the author came to the conclusion that although research methodologies are limited in number, there is a large variety of commercial products. The author has targeted those products that are specifically designed for enterprise modelling. The majority of the tools in this section deal with business process re-engineering. Support of other aspects of enterprise modelling such as promotion of communication and understanding between agents and integration of processes varies according to the objectives of the tool.

3.11.1 Proforma

Proforma provides methodologies and tools for business software development. They build and sell tools for business process re-engineering (BPR), business object analysis, and client/server design and development [Proforma 1999]. The aim of their tools is to perform enterprise modelling, by bringing business and information technology together, in order to define, model, and build business systems. Proforma assists business organisations to achieve business process automation approach. Their The reason for enterprise modelling is business process re-engineering by going through and experiencing all the benefits of enterprise modelling such as business understanding, consistency and communication.

They have identified the need for modelling and integrating business processes as a bear essential for remaining competitive. Their modelling method is based on integration between business process re-engineering techniques with client/server and OO design and development to automate (re)engineered business processes. For this purpose they have developed a tool called Pro Vision. Pro Vision is a software tool for gathering and documenting business processes and requirements prior to application development or software package selection.

The resulting business object models are then used to evolve system design and development. Current project services are:

Business process reengineering (BPR) services to model new or rethink old business processes. The modelling techniques we use in BPR allow organisations to quickly design the most cost-effective and competitive business processes possible.

Business object analysis services to assist joint business and IT teams to fully explore and define their business objects models. Business object models contain all the detailed business rules, decisions, and information needed for system design and development.

Client/server design and development services to transition and extend business objects into system objects and generate application components such as databases, GUI components, and code.

Proforma assists information technology organisations in applying new technology to solve business problems.

An education curriculum which is designed to accelerate business systems professionals' ability to apply business objects to the analysis and design of client/server solutions within a network of legacy applications.

Consulting and mentoring support to assist organisations in planning for and implementing new technologies such as BPR, object orientation and client/server.

A business process and object modelling and automation tool called ProVision Workbench ProVision Workbench seamlessly integrates business process reengineering / BPR, business object analysis, and client/server and OO development. It provides a common repository for modelling both business practices and information technology requirements.

3.11.2 Software Data Environment

Software Data Environment (S.D.E.) is a software tool for modelling environments of software development companies. Unlike Provision mentioned above S.D.E. is dedicated to understanding business processes making forecasts and running 'What if' scenarios. It consists of a number of integrated design models designed for organisations developing products and providing services to the software industry.

The tools comprises enterprise models, data warehouse model, subject area logical models and subject area application models. According to the manufacturers/ developers of the tool S.D.E. provides an integrated family of design model products that specifically address the planning, marketing and fulfilment requirements of the software industry [SDE 1999].

They also claim that the development and marketing of software products with accelerated time-to-market and shortened product life-cycle requires the co-ordination of each functional area of the organisation. The Software Data Environment integrates the information needs of the entire organisation to support product planning.

marketing, forecasting, pricing and distribution. They have concluded that software companies in general have a critical need for integrated planning and reporting of information. However, the nature of the industry precludes many companies from taking the time and resources from new product development to complete this work.

S.D.E. is designed to address this issue and provide a solution from which this work can be quickly completed by taking advantage of a baseline of common information practices in the software industry. Some of the features of the tool are software forecast, product upgrade and version control, problem reporting, sales analysis business analysis, inventory and profitability analysis

3.11.3 BPR Top-ix analyser

This is another software tool dedicated to business process re-engineering. Top-ix analyser provides a method not only for the initial BPR process but also a programme of continuous improvement thereafter [Top-ix 1999] .

According to David Willis [Top-ix 1999] BPR is about achieving the highest customer value from your business processes. This is not a static position as customer needs and competitive environments are changing continuously. It is not enough for a company to complete a successful BPR project, it must continually revisit its process to identify where more value can be added.

Top-ix treats reengineering as a logical 'journey' through discrete stages which must all be successfully completed if the project is to work. This takes a top down approach, moving from high level business strategy to the assignment of individual tasks and project implementation.

The first stage employs high-level process mapping to give an organisation a clear understanding of how its business functions. By aligning the requirements of an organisation's customers to the requirements of the business itself, this stage identifies where 'quick hit' process improvement can be made.

Once the first steps on this reengineering journey have been taken, an organisation can 'drill down' to map its sub processes involved in any process and their respective

activities and tasks. This gives an overall view of how the business functions at the moment and allows a company to move easily to the next stage of process evaluation and design.

This stage allows for the evaluation of existing and future processes against a wide range of criteria, including:

Value-added analysis
Customer service times
Delivery cycles
Quality service times
Activity-based cost standards

Once new processes have been designed and modelled, the journey moves onto the implementation phase by starting to design the jobs associated with the processes. Optimisation of resources; product or functional costings, organisational modelling and staff development can all be planned for here. This ensures the smooth implementation of new processes or improvement to existing ones.

"Using a structured method, such as the Business Process Reengineering Loop, increases the changes of successfully reengineering dramatically. By breaking the entire task down into easily understandable and attainable stages, TOP-IX's approach removes the confusion and inertia which has been associated with other BPR projects," according to Managing Director [Top-ix 1999].

3.11.4 Pangaro

Organisational modelling is the capture of a shared mental model of an organisation of any type, be it a large corporation, a small business or a work group. The model is primarily concerned with the goals of the organisation (stated or implied), the activities of its parts, and the responsibilities of the individuals involved [Pangaro 1999].

Although software houses have developed a number of tools to support enterprise modelling aspects. The company is engaged in systems consulting, and software

research and development in areas such as decision support systems, intelligent training and knowledge-based systems, cybernetics and artificial intelligence, object technology and its associated methodologies, groupware (software for collaborative work), modelling and simulation, computer graphics, and user interface.

They also claim to specialise in techniques for analysis and methods of transformation for organisations of all kinds, strategies, information management infrastructure, environment,

They also carry out evaluations of information infrastructure strategies, in the new context of commerce, systems modelling, software evaluations, rapid prototyping and demonstration software.

3.11.5 Others Enterprise modelling tools

BDF is a software tool developed by Texas Instruments Inc [Texas Inc 1999] and it supports all aspects of enterprise modelling such as Process Modelling, Organisational Modelling, Data Modelling, Flow Modelling, Model object interaction using matrices, Simulation and evaluation of existing and proposed designs, Information Engineering support, IDEFO and IDEFIX support, Model customisation Model repository.

3.11.6 System Architect 2001

System Architecture is an integrated package of modelling tools and is inherently advantageous to both the user, who receives a more tightly integrated tool system and therefore excellent consistency as well as more tools for a minimal cost, and to the vendor, who only needs to utilise and package a single code base. The architecture discussed above can easily be compared with Microsoft Office or Visual Studio products, where it is more practical and cost-effective to buy the whole suit or package, rather than just a part of it.

System architect 2001 is not just a collection of are a different tools it is a fully integrated system, sold as a single product, with no alternative configurations or add-ons. Previously distinct tools have been redesigned for structured methods, business, object and dictator would link, interview a single new repository and code base.

It provides support for business, object and data modelling, but it also provides a alternative options for each entity. If we look at Business Process Modelling (BPS), System architect 2001 can support IDEF0, IDEF3, and IDEF1X diagrams as well as computer science corporation (CSC) catalyst framework, with tools that can capture and model business processes, organisations, applications, and technology.

Numerous alternative approaches for analysis and design are supported by System Architect 2001. The traditional structured techniques, such as Gane & Sarson, Yourdon/DeMarco, Ward & Mellor, Information engineering as well SSADM, are supported by System Architect 2001, which also provides for Object Modelling approaches with essential set of UML 1.1 diagrams, plus Shlaer/Mellor, Coad-Yourdon, Booch and OMT. The fact that System Architect 2001 is capable of supporting both classical and modern approaches in one tool, makes it easy for organisations to re-use existing models during transition.

Lastly, a set of data modelling tools provide logical entity/subject area and physical database modelling.

Various alternatives are provided and are able to support differing skills and individual preference, as well as recognise that no single approach is yet comprehensive enough to meet all requirements. By using a combination of various techniques it may provide additional information, as yet untapped by single models alone.

Despite the fact that System Architect 2001 is a comprehensive set of tools each individual tool can compete the top of the range contemporaries but it is important to recognise that system integration provides more than the sum of the individual components. There are many levels of integration including horizontal integration, which takes place between tools on the same layer of the SDLC, vertical integration, which occurs across layers, and development co-ordination across developers and projects. It must be noted that vertical integration with downstream stages of SDLC is not as comprehensively compatible with System Architect 2001, in fact compatibly is limited to data concepts. Integration does not simply take place across UML diagrams

but in System Architect 2001 also has integration across the tools, for example all data concepts are consistent across BPM, Object Modelling and Data Modelling tools.

Comprehensive development co-ordination is supported by a repository and Active Diagramming, a shared repository allows different projects to share models while Active Diagramming provides access to the model by many users and updates are performed in real time. Both the repository and Active Diagramming will be discussed more fully.

3.11.6.1 Market Position

The initial aim was for a CASE tool to be on every developers desk, and although the aim has yet to be realised it can be achieved by ensuring that the modelling tool is comprehensive, high performance and competitively priced.

Popkin has focused upon horizontal integration covering a broad modelling scope, while rivals have focused on more vertical integration with essential tools of systems development such as coding, testing, or configuration and project management. Both approaches are sold by the same vendors, and it is a choice between consistency among a large number of developers in different project's through a common modelling tool versus consistency throughout the life cycle. Both approaches are very useful but at present there is no ideal solution.

Unlike its competitors Popkin has grown its products together, using a common architecture over a period of 10 years, thus resulting in a highly esteemed and well integrated product especially in the areas of SDLC that it covers.

There has been a great deal of success recently with many customers ordering $100,000 worth of tools, these customers include IBM, Airtours, Seimens and AT&T. The platforms used include Java, C++, Microsoft visual basic, while the database to supported include Microsoft SQL Server, PROGRESS, AS/400 and many more. The import export capabilities of system architects 2000 include UML the import and export via Microsoft repository and export to CACI simulation, as well as CASE interface format (CIF).

3.11.6.2 Technical features & Architecture

The system architect 2001 marks a change in the trend of manufacturing different solutions to satisfy individual markets, this architectural integrates all the modelling variants for BPM, OO, to modelling into a single tool with a shared repository. The architecture is fully flexible and can be customised by administrators, for example the features available to individual developers can be restricted. It has a 32-bit architecture and it has a completely modern, Windows-savvy application. Capabilities can be increased by using different components for example, use of Microsoft distributed component object model (DCOM) which enables the repository to be used by developers in real time over a local area network (LAN), and while provision of VBA allows extension of the capabilities of the tool set.

3.11.6.3 Usability

System architect 2001 has an Explorer-type interface that provides the all the advantages of the windows style browser. System of architects 2001 improves on the previously used dialogue boxes in the following ways; on-screen editing allows direct entry of text into diagrams, a new editor matrix provides a more efficient way a data entry than the traditional dragging and dropping of icons icon and it is also a useful visualisation tool.

The diagrams in System architect 2001 are fully customisable allowing different shapes, lines and colours to be selected as well as optional icons and symbols. Whilst most of the visual appearance in diagrams is customisable, interactive or on demand rules checking still enforces the vigour and completeness of the information modelled. This may allow organisations to develop their own methods, however the Butler group would favour less diversity in notations in diagrams to improve communications between companies and reduce training.

3.11.6.4 Business Process Modelling tools

A number of tools are currently provided by System Architect 2001which enable information about the business to be captured using business terminology. This improves both communications between business uses and the IS Department and ensures that business terminology is reflected correctly by information systems. The

complete range of techniques for BPM includes IDEF0 (function modelling), IDEF3 (process modelling), IDEF1X (data modelling) and CSC (catalyst) and many others.

Conventional techniques are supported by a set of information engineering like the diagrams for modelling organisation's hierarchy asked, location information, paper models and functional/procedural modelling. A comprehensive profile of the businesses can be compiled through textual dialogues that contain a diverse amount of business information, such as customer profiles, product profiles, organisational values and critical success factors. The same tools are used at the core of System Architect as are used by BPM and CSC's Catalyst project management. The IDEF0 models decisions, actions, and activities of a system. The IDEF3 allows process flows to be modelled and if exported to CACI's SIMProcess tool it is used for simulation and animation. IDEX1X provides conceptual data modelling which used through System Architect 2001 schema generating tools, can be transformed into physical data models and can generate database definitions.

3.11.6.5 Object modelling Tools

Successful modelling tools have to be able to adapt and progress rapidly to support UML. UML is a standard notation for the way object concepts should ideally be presented through the context of nine specific diagrams. System Architect 2001, unlike other solely UML-based modelling tools, it continues to support other object modelling techniques (OMT, Booch, Coad-Yourdon) and notations.

As previously discussed tool integration allows access to a common repository information such as data structures and elements from both object and structured technique diagrams. As well as the capabiliuty to transform Entity Relationship Diagrams to UML class Diagrams and back, which allows consistent representation across all diagrams.

It is expected that System Architecture will develop with the market and be responsive to its needs and it is expected that in the future improvements to components modelling support will be forthcoming, to support new component model techniques.

3.11.6.6 Structured Technique Modelling Tools

It is well known that neither Object Modelling or object technology [Udell 1994] have conquered the analysis and design market, many developers still develop their own system using classic structured techniques alongside the inventory of modern systems using popular approaches. System Architect 2001 caters for a diverse range of structured techniques including, SSADM (important for the UK), Ward & Mellor (important for real-time arena), and Information Engineering techniques (which are the most used). The structured techniques discussed above, along with Gain & Sarson and Yourdon & DeMarco, use numerous methods such as Entity Relationship and Structure Charts and are all supported by System Architect 2001.

Tool Automation provides a lot of beneficial features such as; DFD's can be automatically levelled, decomposition Diagrams can be generated to highlight the hierarchy of users levelled set of diagrams and the perspective of either the child diagram or the parent symbol can be vertically balanced. As well as Rule Checks, which are used in order to determine missing definitions, unbalanced flows and processes as well as missing data.

3.11.6.7 Data Modelling Tools

Generally modelling tools are used for database design and data modelling (due to the finite nature of sets of rules and variations), however DataBase Administrators (DBAs) use these tools to automatically transform logical data concepts into physical databases. Data modelling is fully integrated within System Architect 2001, thus consistency with data concepts in object models and structured diagrams or BPM can be achieved more easily.

System Architect 2001 separates logical data modelling and physical data modelling, which is essential because logical models are normalised to minimise data redundancy while physical models require to be de-normalised in order to maximise performance. This distinction is also helpful because the same logical data concept might be implemented differently in dissimilar RDBMS.

3.11.6.8 Data Modelling Capabilities

The Data modelling capabilities of System Architect 2001 comprise entity relationship diagram, physical data model, referential integrity constraints, user-defined data domains, automatic propagation of foreign keys, physical data model generation from ERD, IDEF1X and OO class diagrams and reverse engineering of databases into ERD.

The System Architect 2001 ERD is utilised to deal with the logical side while other features are incorporated to assist with and automate the physical implementation of databases.

Schema generation can transform ERD, IDEF1X and OO Class diagrams into data models. System Architect 2001 has a wide range of properties essential to describe a physical database implementation and can be modelled and generated using numerous data schema generation capabilities including; Data Definition Language, physical storage requirements, primary and foreign keys and so on. System Architect 2001 provides a number of generation capabilities including the few mentioned above, these allow utilisation of a useful range of databases, in addition existing database definitions can be reverse engineered into an ERD.

3.11.6.9 Code Generation and Round-Trip Engineering (RTE)

System Architect 2001 is able to generate codes from models for a number of languages, which in some cases can be reverse-engineered, the process which is called RTE, to return back into a tool, however the generation and RTE capabilities are limited to data concepts solely. A diverse range of object and 4GL languages are supported and C++ headers can be reversed into a class model, and thenm regenerated into Java, Forte or Dynasty. RTE capabilities and language generation are available to a number of software packages including C++, Microsoft visual basic, Magic and so on.

3.11.6.10 Development Co-ordination

The repository, which has previously been discussed, is at the heart of System Architect 2001 and supports the notions of hierarchical/corporate/project/individual repositories although the structure is not compulsory. Specific model objects can be

locked by developers and can be modified using a check-in/check-out approach, or make it fixed in order to make it impossible to modify. Mergers can take place between models and facilities to control the impact of change and reconcile difference, versioning may also take place at the repository level.

3.11.6.11 Active Diagramming

This provides the opportunity to share a repository between many users along with real-time updating, which is especially useful to projects and workgroups of the number of 10-20 concurrent users. Other capabilities also enable consistency both within and between teams and multiple alternative views can be constructed, each using the same repository, which is useful for focusing upon a specific part of a complex model.

3.11.6.12 Extensibility

The repository is extremely flexible and extendable especially when used in conjunction with Microsoft Visual Basic. It allows new objects to be defined and assigned appropriate tools and diagrams, as well as the ability for the user to define his/her own matrices allowing speedy data entry and an alternative to dragging and dropping icons. VBA has been added as a full scripting language allows access to the full object model of the repository, and permits new objects to be defined and rule checks to be carried out to assess their completeness and integrity, with the possibility to define whole new diagrams.

3.11.6.13 Reporting

Some 150 standard reports are currently provided, as well as a graphical SQL-based report designer. Reports can be produced as HyperText Markup Language (HTML) allowing easy access without a tool, yet it lacks a dynamic nature, being a fixed report unable to update in real-time. Templates for Microsoft Word are provided to enable custom formatting of reports.

3.11.6.14 Future

Major upgrades to the system include updates to OO and Business Process. OO update will included amended component support and Round-Trip Engineering (RTE), while Business Process updates include more usability of features for model

navigation and customisation as well as enhanced method support for enterprise architectures.

3.11.7 Logica developments

The company Logica has developed a model called CADDIE [Logica 1992] which can simulate an organization's environment and assess the impact of I.T. on the current organizational structure. This system draws on the latest advances in open systems, concurrent object-oriented programming, and organizational theory. An important part of modeling organizational structure is to recognize that it is both formal and informal, and changes in response to the external environment.

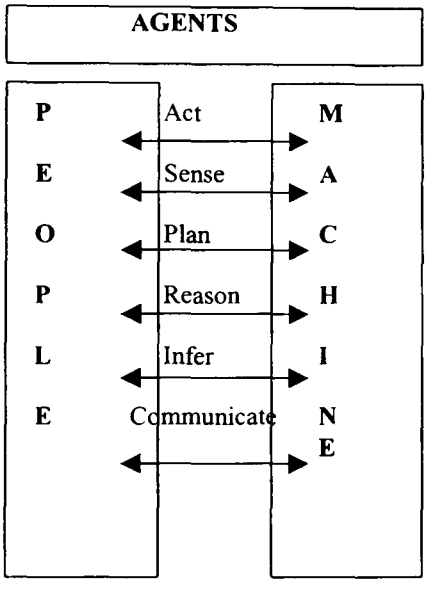

Figure 3.12 CADDIE principles

In the CADDIE architecture both people and machines are represented as 'agents' who possess characteristics such as the ability to sense, act and reason, plus the capabilities to plan, infer, communicate and negotiate. The agents can be grouped into social and functional groupings which can operate in simulated and real worlds. Logica have implemented *CADDIE* using *C++*. This was to ensure that it is portable and suitable for integration into large scale, open commercial applications.

To assist this development Logica has used the ROCK (Representation Of Corporate Knowledge) software from the Carnegie Group Inc. ROCK is a C++ library and set of utilities and has been used for the advanced knowledge representation of agents, domain tasks, event categories, agent beliefs, uncertainty, constraints and controls.

3.11.8 InterTrans Logistics

The company InterTrans Logistics focuses on the transactional, operational, tactical, and strategic aspects of transportation and logistics activities across the supply chain. Their products support many of the basic functions of supply chains such as information handling, and decision support capabilities. They have developed a range of supply chain modeling products. Some of the most popular products are presented below.

The *Supply Chain Strategist* [InterTrans1998] is a network modeling tool which brings some similarities to the tools developed during this research. The tool supports a detailed *'what if'* analysis across a supply chain network. It allows users to model product flows and their associated costs, capacities, and service constraints; from raw materials through to production, distribution, and consumption. Alternatively one can focus on individual supply chain segments such as component sourcing or customer distribution. This allows managers to understand the total cost, profit, and service trade-offs that exist between alternative network scenarios.

The *Carrier Bid Optimizer* [InterTrans1998] is another tool by the same company. The tool provides strategic decision-support that enables shippers and carriers to collaborate. Shippers can analyze, forecast, and format transportation needs into a well-defined set of requirements that enable carriers to make conditional, sequential bids on a set of well-defined lanes, and selectively price according to their efficiencies and operating networks. Finally shippers can efficiently allocate lanes to carriers, while expediting the carrier selection process, and benefit from improved service at lower cost.

The *Transportation Optimizer* [InterTrans1998] is a third tool produced by InterTrans. It is also a decision support application responsible for analyzing alternative transportation planning scenarios for large, complex shipping environments. The tool uses real world data to automatically consolidate shipments, optimize delivery routes, calculate actual carrier and transfer costs, select carriers and pool points; all with a view to minimizing overall freight costs. InterTrans have also

developed two more tools which are less relevant to the scope of this research both dealing with transportation routing and scheduling.

3.11.9 IBM-BPMAT

BPMAT is a supply chain modeling tool developed by IBM. It is a software tool for modeling and analyzing business processes that combines static process modeling with activity-based costing and simulation. The advantages of the tool are that consultants can evaluate alternative business process scenarios using simulation and then select the most promising, based on quantitative metrics such as cost, cycle time, resource utilization, throughput and serviceability. BPMAT is graphical tool that can be applied to any industry. Models created with BPMAT can be stored in libraries and are reusable. In addition the tool supports the development of models which represent the inter-related flows of products and information among suppliers, manufacturers, distributors, retailers and customers. IBM claims that the tool can reduce the cost of business process reengineering, increase quality of recommended process changes, quantify the potential benefits of change and reuse intellectual capital in the form of process libraries.

3.11.10 NCR Development Group

A rather more interesting (due to its multiple capabilities) software modeling tool was developed by the NCR Enterprise Model and Development Group. The tool is capable of offering modeling solutions to different levels within an enterprise. The multilevel logical representations of the enterprise can be viewed graphically. It can be used to model any level of an enterprise: strategic level, business process level or activity (workflow) level. It can also model processes, product or goals. The tool supports different views, such as people, information, geography, relationships and schedules as well as allowing multiple views of the dynamics of an enterprise in an 'enterprise cockpit'.

3.12 Relevance of E.M. Methods & Further Discussion

If we focus upon the architectures and modelling methods discussed above, we can see that almost all architectures relied upon numerous perspectives of manufacturing enterprises. Most of these architectures have differing views however the functional

and informational views appear frequently. These differing views can be seen if we look at a range of architectures and modelling methods already discussed. IDEF has a dynamic view, while CIM-OSA architecture takes a resource and informational view. CAM-1 uses three additional perspectives; the management prospective, computer systems perspective, and the physical structure perspective as well as the functional and informational perspectives. An enterprise can be modelled using components composed from different views, aspects, and perspectives.

It has been proposed [Weston 1994] that there are three E.M. approaches to CIM. These are engineering perspective, information systems including networking and operations. Manufacturing organisations are encouraged to investigate a CIM in the context of a total organisational perspective. The single dimensional CIM approach will produce results below expectations, especially in a global competitive environment.

Other authors [Brandimarte 1995] have pointed out that the aim of the CIM community is to produce an integrated manufacturing enterprise yet the reality of integrating different domains and cultures is very difficult to achieve. Despite these difficulties the aim of integration is advantageous to the community especially operating upon the physical, communication (information) and operation dimension.

[Little 1991] states that it is false for all systems analysts to believe that they should be building complete models of the system and setting of the control variables. If that were the case major opportunities to improve the system and finding new ways of improving the people at the front line of the organisation by providing a information, training, and tools. [Little 1991] goes on to say that using the same methodologies to build more complex models is likely to yield the less results. He argues against a single integrated model claiming that a hundred different models are often required. The of impracticality of a single model can be exemplified by the fact managers face hundreds of different problems daily such as; employee absenteeism, products, customer dissatisfaction and so on and thus would require many different models not just one. This is a very strong argument opposing to the building of a single integrated model.

The following section will deal with the differences, if any, between the development of an integrated model and model integration. The former is the development of a holistic or global view, adopting a top down or hierarchical approach, from which a localised model can be built by 'zooming in'. Most of the models previously discussed fall into this former category. Model integration is the pooling of different models and linking them with some commonalities. The advantages of the first approach (integrated model) is that local models created with holistic picture in mind will not be striving towards perfecting local models and would contribute toward system effectiveness. However the disadvantages to the first approach is that it would be easy to adopt while establishing new systems but it would be a huge task to use this concept in existing systems. A big challenge faced by designers is to make sure that these models are flexible as well as being asked to adapt to changing environments without losing focus.

In support of the second approach (model integration) is that it is a practical integration methodology especially with reference to existing organisations, however the problem is that each of the local models is the developed separately and will naturally strive towards local model perfection. Other inherent problems include how to make these models communicate effectively with each other, especially as often models are developed using different languages and on separate platforms.

[Heim 1994] presents model integration using ENVISION architecture, which attempts to integrate distributed models. Model integration is an option or alternative to aggregate refinement and decomposition. Model integration joins individually and independently functioning models to create more complex and comprehensive models that can share data and co-ordinate the modelling activities. Designers are able to use information from existing models to reduce programming effort, simplify model validation and increase scope of the design option considered in the project time available.

The approach of integrating distributed models built using different tools to represent different views is challenged by [Wang 1993] who cite [Gay 1991]. Gay describes the functional, informational, and dynamic perspectives by applying IDEF0, IDEF1

and SLAM11. These approaches are challenged by [Wang 1993] using the following arguments:

Different environments have to be used to build representations of different viewpoints, some activities have to re-represented in building the dynamic version of the model, all of the this takes a lot of effort to complete the whole process.

Different environments have different principles and syntax, this makes it easy for information items to get lost especially when different people have been involved in building their representations.

Another problem is that when modifications have to be made to the functional model and the information the model, the relative modifications and the dynamic model cannot be done automatically. This makes the whole design and modelling process even more open to errors and a very inefficient.

[Wang 1993] proposed a solution to the problems they discussed above, a comprehensive methodology called IDEM (integrated system description model). This methodology is developed by modelling a system from three connected to but differing viewpoints in the same context. Each separate viewpoint captures certain important aspects of the system but all three are required to provide a holistic description. Initially a functional representation of the system is built and then the information view of the system is added on using an object-oriented information modelling approach. The clarity of information objects may be produced and associated with the items in the functional model. The temporal attributes are specified for the functional model, and further procedures to govern the dynamic nature of the model. Rule sets are utilised to express the decision-making know-how that governs the dynamic process. IDEM is produced using LOOPS, an integrated knowledge engineering language encompassing all objects, regulations, access and procedure oriented programming capabilities.

One of the general conclusions derived from studying enterprise modelling methodologies and tools is that dynamics as a concept is proportional to the scope of the model. Models that cover a wide area of an enterprise tend to offer little or no

animation capabilities whereas models that target a specific aspect tend to be more capable of accommodating and dealing with change. Often they tend to 'forget' the fact that enterprises are open systems, and that external changes may affect their environment. Some of them also fail to see the 'big picture'. This is something that has been acknowledged by enterprise modelling researchers while some methodologies offer modelling capabilities of the wider area of the enterprise. If one used Morgan's metaphors [Morgan 1986] to describe the enterprise modelling approaches suggested so far, one would realise that the view of the models is mechanistic. Organisational structures are broken down into goals supported by businesses which are then supported by tasks. There are surely strengths with this approach such as; businesses are broken down to straightforward tasks that are easier to perceive and understand. It does require however, this mechanistic approach, a stable environment, precise definition of goals and as Morgan puts it 'the human parts to comply and behave as they have been designed to do'. There are however a number of drawbacks. It many cases it leaves no room for experimentation or exploitation of alternatives. The life span of these models is usually very small. This mechanistic approach creates models that are difficult to adapt to changes and are therefore discarded. Morgan also claims that mechanistic views of enterprises may promote predetermined goals but they lack innovation. There is however a great deal of lessons to be learned from the modelling methods examined so far.

Ontologies and the TOVE model propose a structured approach to enterprise integration. The success of the approach lies in the quality of the ontology and how well it can support communication. The construction of ontologies is not an easy task. It requires good knowledge of the enterprise's operations and resources. The terminology used must be consistent, otherwise the communication protocol will produce ambiguous results. Consistency and adherence to definitions has to be ensured so that the ontology enables knowledge representations to be communicated between departments.

On the other hand, methods for computer integrated manufacturing environments also require detailed knowledge of the activities of an enterprise. In order to develop a functional view of an enterprise using CIM-OSA, the analyst must have good knowledge of the domains, domain processes, enterprise activities as well as the

business processes. Furthermore, in order to develop the model up to the information view level, additional information related directly to tasks, resources, time and labour is necessary.

The approach taken here deals with the problem of developing and animating models by manipulating the nature of their relationships. The difficulty of this task lies in the fact that one could not create a generic definition for these relationships, as they depend upon the nature of the enterprise. The most essential factor that needs to be taken into account, is the strength of the link. This is usually determined by political factors, such as the power of the enterprise, who initiates the link, or the degree of dependency between the two parties. The result will be a network of enterprises. The model also attempts to assess the impact that creation or deletion of a link has on the rest of the network. This is done in terms of what the alterations are, and what the position of the enterprises will be after the link has been broken or created.

The model considers the relationships between enterprises and has the ability to predict the impact of enterprise creation or deletion. The thesis is not directly concerned with the internal operations of an enterprise (product manufacturing or distribution), but with the relationships between enterprise components.

Ontologies are used to enable communication between departments within an enterprise. However, when we refer to links between enterprises as opposed to links between departments within an enterprise, we really refer to confidential information. Most large enterprises would be reluctant to share such information with analysts who are trying to model network dependencies. This wouldn't be the case within an enterprise, hence common terminology and definitions can be produced. Therefore I have taken a different approach to model these links and to predict the impact upon enterprise creation or deletion.

3.13 Conclusion

The chapter presented current research in the area of enterprise modeling, as well as products currently available on the market. During the research stages into the area of enterprise modelling it became apparent that enterprise models deal with a number of

issues from organisational modelling to enterprise integration. Depending on their objectives some models approximate or simulate an enterprise environment for the purpose of re-engineering or BPR while other consider a wider view. The thesis acknowledges that there are a number of enterprise types not all of which can be modelled with the methodologies examined in this chapter.

Methodologies such as CIMOSA, PERA, ICAM and WADE have been developed for CIM environments. IDEF is a generic method for process modelling and therefore can target several distinct types of enterprise. Business process reengineering seems to be one of the main concerns of many of the methodologies we examined. The process of BPR can take place outside a computer integrated manufacturing environment too. Although many of the methods examined in this chapter aim in facilitating the BPR process within a CIM that does not imply that BPR and CIM are inter-linked. The same guidelines and principles for carrying the task of BPR would apply to any enterprise regardless of its type. NBS is a generic architecture for modelling enterprise entities and it follows a hierarchical structure such as factory-shop-cell-workstation-machines that can be applied in a number of instances. Tools such as SSADM (structured systems analysis and design method) and SSA (structured systems analysis) can also be considered generic since they enable the modelling of data flows or the flow of physical units within an enterprise. Finally the concept of object orientation and design has been widely applied in several types of environment for the modelling of data or physical entities and can therefore considered generic.

The main purpose of supply chain models for example –although they are built from an enterprise point of view- is to assess one's market position and assist the process of optimisation. Another distinction can be made between static and dynamic models. Static models display the picture of an enterprise or supply chain as it is perceived at the moment of their creation. Dynamic systems on the contrary allow the model to animate in order to accommodate changes that affect the initial picture. It should be stressed here that although static enterprise models target a number of aspect of an enterprise such as resources, processes, time scales etc, dynamic models target a particular aspect such as transportation scheduling [Scott 1995]. In fact the majority of dynamic models found during this research deal with modelling of logistics.

Enterprise modeling is a broad area and receives attention from many academic disciplines. The area of particular interest in this research is communication between the departments of an enterprise or between enterprises themselves. The chapter presented a series of research methods and tools to achieve these goals. The way these *links* (either internal or external) are perceived has a great impact on the enterprise's operations. Having established a clear picture of the methodologies and techniques for enabling communication between departments within an enterprise, possible ways of modeling links between enterprises and how they affect the decision making process have been considered. The later chapters of this thesis examine the importance of these relationships, how they can be modeled and their impact on an enterprise's activities.

Chapter 4 Dynamic Method

4.0 Introduction

The previous chapters addressed the modelling problem in detail, and demonstrated the need for a modelling method that would be able to offer animation capabilities to an enterprise model. In a dynamic social network, changes caused by internal or external factors play an important role at a tactical, operational and strategic level. This chapter outlines the steps that need to be taken in order to build such a dynamic method and discusses the relevant issues. It proposes a method for modelling relationships of dynamic social networks and although the method is a four-step approach the chapter also discusses a number of issues that led to the development of each step.

4.1 Problems with Current Methodologies

Modelling the relationships between enterprises or enterprise components is an extremely complex task due to the vast amount of information involved as well as the different levels of impact it may have. There are a number of issues related to modelling and animating relationships between a network of agents. The most important issue is deciding where to draw the boundaries. This implies the boundaries of the model itself as this has to be decided before trying to develop and animate. Introducing or removing an agent from a network of interconnected agents has different levels of impact in the model. When the model for example is an enterprise and the agent is a new activity, this can impact all levels of operation (strategic, tactical, operational). Enterprise modelling methodologies like CIM-OSA and PERA provide frameworks for the modelling of activities, resources and labour within an enterprise. Their primary objective is to model the functions in an enterprise and the resources required for each function. The majority of these methodologies were developed with a computer integrated manufacturing environment in mind. Their success lies in their ability to develop models of several viewpoints (functions, resources, labour) as well as the interrelationships between them. In the case of CIMOSA the models reflect an enterprise from two different viewpoints; the functional view and the manufacturing view. The former models the tasks from a data

flow perspective and the latter from a resources perspective respectively. The first view deals with information and data that are relevant for the performance of each task, whereas the latter view deals with the equipment and the people related to each task (function).

However an enterprise can be affected by more than a set of activities, resources and people since it also consists of a set of links with the outside world. These links usually take the form of dynamic relationships as they are established under a very competitive environment. The information that is passed to an enterprise through these links, as well as the information that is passed to others, has a direct affect on the way an enterprise operates. The same observation can be made for supply chains. We cannot argue that a model is accurate unless the data, information and knowledge which travels through these links has been taken into account during the modelling process. Consider the automotive sector of a regional economy.

If for example a new tyre manufacturer is introduced in a region and this type of product was missing from the sector it will impact the automotive model in several ways. The tyre manufacturing company may benefit financially since it may be able to take advantage of lower delivery costs, hence a financial impact. Assuming the tyre manufacturer can produce enough capacity to supply the car manufacturers of the model then the car manufacturer may also enjoy financial benefits since they will save in import costs. Assuming the tyre manufacturer employs a number of people this also implies sociological impact. However the tyre manufacturer's plant during the production stage will release a certain amount of waste and this is has also an impact in the environment. Now consider the tyre manufacturer leaving the region. Again affects will be realised throughout the different levels we have just mentioned. The above example demonstrates how adding, removing or even evolving relationships can have different levels of impact. The method proposed in this thesis provides a methodology for developing and animating enterprise models or networks of interconnected agents as they are sometimes refer to. This chapter describes the nature of the approach and how it can be used to model

relationships between agents of enterprise networks. Special attention will be given to 6 key points, which are, in order of appearance:

- The need for modelling environments as open systems
- The ideas behind the approach and artificial intelligence planning
- Modelling agent relationships using a thesaurus based approach
- Conversation theory to quantify relationships
- Representing a model using lists
- Processing those lists to animate the model

The data structures for representing a model, processing its state and animating it resemble Lisp type data structures. This particular representation was used because of its clarity and also because lists data structures have been widely used in AI planning. Although they resemble Lisp data structures closely there was no Lisp compiler used. The author refers to the *automotive sector* throughout the chapter, although the approach suggested could be applied to other sectors of both service and product manufacturers. In the following paragraphs we discuss a number of issues related to modelling enterprises as open systems. An important conclusion that was drawn from the literature review is that modelling in general-regardless of its purpose- could start as early as the systems development life cycle. Systems that have been designed as living organisms for example, are found to embed a number of elements which could assist or even guide the development of an enterprise model. If the dynamics concepts of the enterprise have already been considered since its development, enterprise modelling could use them as a vehicle rather than re-invent them as it is usually the case. The following paragraph describes the notion of open systems and gives a detail account of the dynamic concepts or elements that could be embedded in a system during its development cycle.

4.2 Open Systems

An open system is a unit that is dependent upon the wider environment to fulfil its needs. It has long been acknowledged that all units regardless of whether they are individuals, groups, and organisations have needs to be satisfied. Thus organisations like organisms

are 'open systems' in the sense that they are 'open' to their environment and it is essential that they maintain an appropriate relationship with their environment in order to survive.

It is this mode of thinking that has been termed the system's approach to organisation. This approach has evolved from and it is based upon the work of biologist Ludwig Von Bertalanffy [Bertalanffy 1969]. This approach was developed simultaneously on both sides of the Atlantic during the 1950s and 1960s.

At a theoretical level the development of an Open Systems approach has stimulated many new ways for thinking about organisations. These new conceptualisations are often portrayed as general principles for thinking about all kinds of systems based on Bertalanffy's work 'General Systems Theory' [Bertalanffy 1969] which provided a means of inter-linking many different scientific disciplines. His premise was that a living organism can be a model for understanding complex open systems, and thus understanding the world at large. Initially systems theory developed as a biological metaphor in disguise.

At a practical level, the open systems approach highlights a number of key issues.

The environment in which an organisation exists is of primary importance, although traditional management theorists gave environment very little import and perceived an organisation as a "closed" mechanical system and thus focused upon internal factors. The open systems approach has re-addressed the balance and now environment is placed at the forefront when organisation is undertaken. A lot of attention is now placed upon the immediate "task environment" defined by the organisation's direct interactions; with customers, competitors, suppliers, government agencies and labour unions not to mention the wider contextual environment. A lot of the widespread interest in corporate strategy is due to the discovery that organisations are ultimately connected with and must be sensitive to the occurrences in the wider world. Organisational practice now stresses the necessity of the ability to look for and sense changes in task and contextual

environments. As well as the ability to bridge and manage critical boundaries and areas of interdependence, and developing appropriate strategic responses.

The Open Systems approach defines an organisation in terms of inter-related subsystems. Morgan (1986) uses the following simile to describe systems. Systems are like Chinese boxes in that they always contain wholes within wholes just as the way that organisations contain individuals (who are systems in their own right), who belong to groups and departments, which belongs to larger organisational divisions. Returning to the biological metaphor, if we define the complete organisation as a system, other levels of that organisation will be understood as subsystems, in the same way that atoms, cells and molecules are subsystems of a living organism or unit, despite the fact that they can stand as a complex open system on their own merits. Commonly, systems theorists analyse relations both within and between organisations and use configurations of subsystems to display important patterns and interrelations. A popular practice employed to do this is to look at the central sets of needs which enables an organisation to survive, and highlight the significance of governing relations between them. It is important to recognise the fact that everything is dependent upon everything else in other words everything is interrelated and a change in one variable is bound to influence other variables. This approach was enabled through the socio-technical approach which looks at the relationships between technical, social, managerial, environmental and strategic requirements. This approach encourages systems theorists to find ways of governing the relations between the environment and critical subsystems.

A third focus of the practical use of the system's approach rests in the attempt to establish similarities between different systems, and to identify and eliminate potential problems and incongruencies. Just as the socio-technical approach, which emphasises the necessity to match human and technical requirements, the open systems theory generally encourages the matching of subsystems such as the strategic subsystem, the technological subsystem, the human-cultural subsystem and the structural subsystem which together make up the managerial subsystem. It is here that the principles of requisite variety, differentiation and integration, and other system ideas can be brought into the picture. For

example, the principle of requisite variety is very important in designing control systems or alternatively for the management of internal and external boundaries - for these must encompass the complexity of the phenomena being controlled all managed to be effective. The principle of differentiation and integration is useful for organising different kind of tasks within the same organisation.

Taken together the above ideas have enabled theories of organisation and management to break free of fixed thinking and to organise in a flexible way that meets requirements of the environment. These developments and concepts although beyond the scope of the thesis are usually housed in the perspective of contingency theory and on the practice of organisational development.

4.2.1 The principles of open systems

The principles of open systems developed mainly from study of biological systems and are often used in the analysis of organisations as systems.

4.2.1.1 The concept of an open system

Organic systems at the level of the cell, complex organism, and population of organisms exist in a continuous exchange with their environment. This mutual exchange is essential for sustaining the life and form of the system, since interaction with the environment is the basis of self sustaining organism. It is a common premise that living units are open systems and characterised by continuous cycles of input, output, feedback (whereby one element of experience influences the next) and internal transformation (throughout). The key relationships between the environment and the internal functioning of this system are emphasised through the concept of openness. Environment and system are to be perceived as mutually dependent and in continuous interaction. Biological and social systems with their open nature can be contrasted to the closed nature of many physical and mechanical systems. The degree of openness among systems can vary immensely, some open systems are purely responsive to a selective range of inputs from the environment. An example of a partially open system is a machine that is able to regulate its internal operations in direct relation with variations in the environment. Mechanical or

non-living systems, such as towers, bridges or clockwork toys are closed systems, due to fixed or pre-determined responses. A living organism, organisation, or social group is a fully open system which both affects and is affected by the contextual environment.

4.2.1.2 Homeostasis

The concept of homeostasis (although also a property of a closed system) refers to self-regulation and the ability to maintain a steady state. It is important for both biological systems and organisational systems to acquire and maintain a stable form. Biological organisms require two characteristics; a stable form, and a distinctiveness from the environment while simultaneously having a process of constant interaction. These characteristics are achieved through the process of homeostasis, which regulates and controls system operation on the basis of "negative feedback" which ensures that where deviations from the norm occur they are counteracted and this deviation is corrected. Thus when bodily temperature rises above normal, certain bodily functions are activated to correct the dysfunction i.e., we begin to perspire and breathe heavily. Social systems, like biological systems, also require such homeostatic control processes if they are to acquire and maintain constant form.

4.2.1.3 Entropy /negative entropy

Thermodynamics explain that entropy is a measure of the amount of energy in a system that is available for doing work; entropy increases as matter and energy in the universe degrade to an ultimate state of inert uniformity. Closed systems are entropic to the extent that they are likely to deteriorate and run-down. Open systems, alternatively, are more flexible and attempt to sustain themselves by importing energy trying to avoid entropic tendencies. It is this sense of the term that open systems are characterised by negative entropy.

4.2.1.4 Structure Function Differentiation, and Integration

The relation between the above notions is of central importance for understanding living systems. Generally it is accepted that organisations are a structure of parts and can explain system behaviour in terms of relations between the components, causes and

effects, stimulus and response. However reductionism should be avoided, especially if we look at our knowledge of living systems. It is important to emphasise that structure, function, behaviour and all other features of system operation are closely intertwined. Although the study of anatomy has produced some interesting insights, in order to fully understand such systems requires more than this approach alone. Even the life of a simple cell is dependent on a complex web of relations between cellular structure, metabolism, gas exchange, the acquisition of nutrients, and numerous other functions. The cell as a system is not reducible to a simple structure but is a system of functional interdependence. Indeed, the structure of a system at any one time depends on the existence of these functions and in many respects is only a manifestation of them. The same is true of more complex organisms, which reflect increased differentiation and specialisation a function, for example specialised organs performing specific functions. Thus the operation of the brain which requires more complex systems of integration to maintain the system as a whole. The relationships between structure, or function, differentiation, and integration seen in anatomy can also be seen in social systems as instances of organisations.

4.2.1.5 Requisite variety

The principle of requisite variety is related to the ideas of differentiation and integration. Requisite variety states that the internal mechanisms of a system must be as diversified as the environment with which it is trying to control and adapt. A system is only able to deal with variety and challenges posed the environment by enlisting variety into its internal mechanisms. Any system that cuts itself off from its environment and the diversity it provides is likely to lose its complexity, distinctiveness and diversity and is more prone to atrophy. Thus requisite variety is a crucial feature for all kinds of living systems.

4.2.1.6 Equifinality

The principle of equifinality is embodied by the notion that there are many ways to achieve a desired end state. This principle of an open system is in direct contrast to more closed systems, whose system relations are inflexible and fixed in terms of structure to produce pre-determined patterns of cause and effect. Open or living systems have flexible

patterns of organisation which permit the achievement of an end result using different resources, methods and different starting points. At any given time, the structure of the system is no more than an aspect or expression of a more complex functional process and thus structure does not determine process.

4.2.1.7 System evolution

Evolution is the ability to move to more complex forms of differentiation and integration and thus greater variety, enabling the system to deal with challenges and chances in the environment. The capability of a system to evolve is shown by a cyclical process of variation, selection, and retention of selected characteristics. These are a number of characteristics and issues related to developing, operating and modelling enterprises as open systems that have been pointed out by [Morgan 1986].

4.3 Introduction to the method

Bearing in mind the need for building environments as open system as well as the example mentioned earlier and the multiple levels of impact we're presenting our method into the area of dynamic enterprise modelling. In chapter 2 we defined an enterprise model as a graphical or computational representation of enterprises, aiming to promote communication and understanding of business processes while at the same time provide a framework for assessing changes and making forecasts. In chapter 3 it became clear there was a need for developing models with the ability of running scenarios and recording the effects of these scenarios in formal data structures. One major impact is the relationships between agents as to how these evolve or change depending on the types of relationships introduced or removed from a model. In the following paragraphs we're describing a modelling method for animating enterprise networks. The ideas and conclusions presented in this thesis have been drawn by several paradigms such as artificial intelligence, object orientation, frame based approach to knowledge based systems, thesauri structures and previous work into the area. In a nutshell, AI principles were combined with the enterprise thesaurus structure that held information of networked agents in terms of conversation theory values. All this paradigms are introduced and explained in turn and we discuss the impact they had in writing this thesis. It should be

stressed that although not all the paradigms that were initially thought relevant were used in the final method, they assisted in deriving the conclusions that led to the proposed methodology.

4.3.1 The Animation problem; Does AI Literature give us the answer?

Artificial intelligence literature has taught us ways of generating possibility trees, carrying out planning, and using heuristics. During this research it was concluded that there is a large amount of similarities between the ways we represent problems and making inferences in A.I. and the ways we build and attempt to animate models in E.M. The similarities can be sound in the ways information in stored, retrieved and processed in order to make successful inferences. In AI this is a very important issue since successful planning is heavily reliant on effective inference mechanisms. In enterprise modelling this is a very important issue too. We want to be able to animate models and keep track of the impact of movement but at the same time comply with the rules that determine the direction of movement within the sector the model represents. This can be better illustrated with an example. The example has been taken from [Pratt 1994]. Imagine a robot called Bob who inhabits a house with four rooms; a hall, a sitting room, bedroom and a study. All adjacent rooms have doors connecting them. The following figure illustrates this.

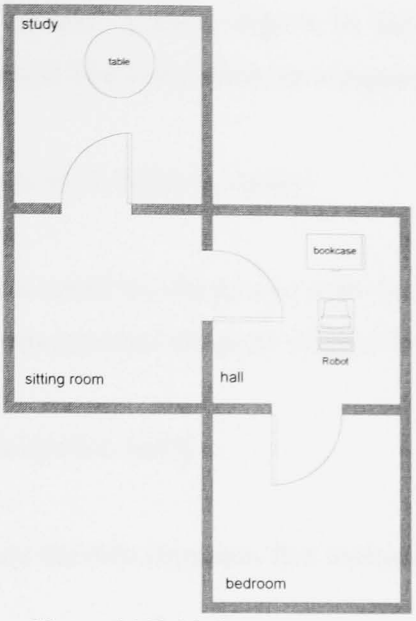

Figure 4.1 Bob's house

One can also imagine bob's house as part of a larger system such as neighbourhood or a block of flats. However the 4 rooms define the boundaries of bob's movement. Bob's function is to move various objects from one room to another. Assume that in the house there are only two objects to be moved; a table and a bookcase. Bob needs to have a programmed plan of the house, together with information as to where the objects including himself are. We also assume that Bob can only perform 2 actions; carry an object from one room to another and going from one room to another. Bob receives and performs scenarios such as 'Put the table in the bedroom, move the bookcase into the study and leave yourself in the sitting room'. Bob first constructs a plan i.e. a sequence of actions he can perform in order to achieve these objectives.

4.3.1.1 Representing the problem

The first step in representing the problem is to device a method is such a way that the computer program controlling Bob can make inferences. Therefore we need to represent the possible situations Bob and his environment can be in, the goals that Bob has been set, the actions and sequences of actions that Bob will have to perform to achieve his goals and Bob's knowledge of the layout of the house. In this example a situation

specifies the locations of all the movable objects in the house – i.e. the table, the bookcase and the robot himself. The initial situation is represented as:

a. [in (bob, hall), in(bookcase, hall), in(table, study)]

This lisp type data structure could be stored and used by the program controlling the robot. In the same way we can represent the goals of the robot.

b. [in (table, sitting-room), in (robot, hall)]

In the same way we represent the data structures that indicate actions.

c. [Carry (bookcase, study, sitting-room)]

Likewise the situation from taken the robot from one room to another can be represented as:

d. [Go (bedroom, hall)]

The plan for the robot is a sequence of actions to be performed in order to achieve his goal. For example, consider the robot is initially in the situation described by figure 4.1, and has been assigned goal c. A suitable plan would involve going from hall to the sitting room, going from sitting room to study, carrying the table from study to the sitting room and finally going from the sitting room to the hall. Again using lisp this can be represented as:

e. [go (hall, sitting room), go(sitting room, study), carry(table, study, sitting room), go (sitting room, hall)]

Finally the representation of the layout to the house needs to be represented. Again using lisp

f. [adjacent (bedroom, hall), adjacent (hall, sitting room), adjacent (sitting room, study)]

4.3.1.2 Representing knowledge about actions

In order to device a plan the robot needs to know the types of actions available to him along with their preconditions. Consider the action type of carrying an object from one room to another. Reasoning for the action depends upon knowing the following:

For any *object* and for any *room1* and *room2*
The robot can perform action carry (object, room1, room2) provided the two rooms are adjacent and that the object exists in room1. If the action is taken and performed then some new facts arise. The robot and the object are in room2.

In order to express this in a data structure, we need to express the type of action, the preconditions as well as the effects. The effects can be considered to be changes to the previous lists a and b.

Action-type (carry (object, room1, room2),
Pre-condition ([in (object, room1),
In (robot, room1)
Adjacent (room1, room2)
Add-list ([in (robot, room2), in (object, room2)]),
Delete-list ([in (robot, room1), in (object, room1)])).

The preconditions are lists of facts that must be true in the current situation prior to the execution of an action. Add and Delete lists are updated to show the effects of each action. Consider the next action.

Action-type (go (room1, room2),
Pre-condition ([in (robot, room1),
Adjacent (room1, room2),

Add-list ([in (robot, room2)]),
Delete-list ([in (robot, room1)])).

These type of lists can be thought of as rules since they control in a sense the movements of the robot. We will be visiting these types of lists throughout chapter 4 and since they contain preconditions, add and delete lists we call them PAD.

4.3.1.3 Making inferences

Representing facts about actions using PAD rules allows the robot to easily compute the results of an action such as,

Carry (bookcase, hall, sitting room)

By retrieving the appropriate PAD rule, in this case,

Action-type (carry (object, room1, room2)

After the preconditions are satisfied and the action executed the new situation that arises is,

[in (robot, sitting room), in (bookcase, sitting room), in (table, study)]

It has been explained so far that data structures can be used to support inferences about effects of individual actions. These inferences could also be chained together to reason about the effects of sequences of actions i.e. plans. Consider the initial situation again,

[in (bob, hall), in(bookcase, hall), in(table, study)]

Now consider the proposed plan,

[go (hall, sitting room), go(sitting room, study), carry (table, study, sitting room), go (sitting room, hall)]

By going through each action we establish that the preconditions are satisfied and then we compute the resulting situation. The sequence of situations is,

[in (robot, hall), in(bookcase, hall), in(table, study)]
[in (robot, sitting room), in (bookcase, hall) in (table, study)]
[in (robot, study), in(bookcase, hall), in(table, study)]
[in (robot, sitting room), in(bookcase, hall), in(table, sitting room)]
[in (robot, hall), in(bookcase, hall), in(table, sitting room)]

The goals of the proposed plan are realised in the final situation. The rest were intermediate states, resulting from actions taken in order to achieve the goals. These calculations can be regarded as inferences and they usually take the form of a hierarchical representation of situations or a tree of possible options.

4.3.1.4 Searching for plans

In order to get a more global view of the planning task we can look at all the possible situations the robot can be in along with the actions that can be performed. As I mentioned earlier these can be imagined as a tree of possibilities. Each node in the tree represents a possible situation, and a link from one node to another below, represents an action that can be performed in the situation represented by the second node. The root node represents the initial situation. Thus, the tree is, in effect, a map of the possible sequences of actions that bob can take in every situation he's in. The problem of finding a plan to achieve a given set of goals can now be viewed in the following terms.

Starting from the root node, generate a tree node by node, testing each node as it is generated to check whether the goals described in the list are satisfied. If one is found then the route from the root to the final node will be the plan for achieving the goal. Of

course the questions here is how do we generate the tree? The tree of possibilities can be generated row by row or branch by branch. The principles of guiding searches in artificial intelligence literature comes under the heading of heuristics. The term heuristics, although it covers a number of aspects of AI research, it also covers planning and finding optimum plans. We examine the notion of heuristics in chapter 5 and show how it impacts enterprise modelling.

4.3.1.5 How E.M. fits into this

Dynamic networks that can take the form of relationships between enterprises in a supply chain or departmental sectors in an enterprise or even components of an activity can be represented and animated using the paradigm explained so far. The above example corresponds to an enterprise in a more or less institutionalised and stable market. In order to help the reader understand this better consider the following parallelism.

The example with the robot was used to describe the situations that could be found within the house it lived in. Situation are described using lisp type lists of actions indicating where is what. If you consider the rooms to be the static objects and the robot along with the bookcase and other objects the dynamic variables, one could say that in a sense these situations represent relationships between static and dynamic agents.

The list type

[adjacent (bedroom, hall), adjacent (hall, sitting room), adjacent (sitting room, study)]

which defines the layout of the house can be considered or thought to be the boundaries of the enterprise model. In our approach this is achieved by using an enterprise thesaurus. The thesaurus is a way of identifying adjacent objects, according to the above paradigm or in other words links between agents.

A PAD rule in the above paradigm represented a series of preconditions. In this approach in order to animate an object within a dynamic network we use PAD rules to show the relationships that can be formed or how relationships can evolve.

The robot used in the story can be replaced by or thought of, as the agent coming into or moving around the dynamic model. The objects the robot brought with it (table, bookcase) can be thought of as variables that indicate the types of relationships it can be involved in or as the restrictions that apply to its movements. We show this by using variables of mutuality and significance, as it will be explained later.

I have concluded that the way we animated bob (regardless of the engineering part) from a situation A to a situation B can be used to animate enterprise models. The physical restrictions of the robot can be thought of as the boundaries of the enterprise model. In order to move the robot from a to b, both situations have to be described and represented in some sort of data structure. This is represented in this method by showing what restrictions apply in the movement of each object in the model. When the new situation is reached then it has to be described again to reflect the changes from situation a as well as the new restrictions that apply.

The movement of the robot in the house is described as a series of relationships between the house objects and the robot itself. The enterprise model objects depending on the type of relationships that they are in (which is expressed in terms of mutuality and significance) can form a set of relationships that can be pre-determined.

Every situation changes the set of options available for animations. In the case of the robot if it's in its furthest place it can only move in three directions. In the enterprise model if an agent is engaged in an asymmetrical competitive relationship then it cannot evolve to a parent child relationship. In order to perform an action to the model or the robot that is not adjacent to the current situation then we need to execute a plan. In order to execute such a plan we need a global view of the planning task by looking at the collection of the possible situations together with the actions that can be performed in

them. The various situations the robot can be in which are expressed in lisp type form are also expressed in lisp type form in this approach. Although in the first case they show the rooms the different object can be found in, in this approach they represent the relationships between agents. AI principles were used to animate and keep track of changes and other information as it will be shown later in the chapter. Prior to animating however we needed to develop an initial picture of the situation. The following paragraphs explain how object-oriented design was used to represent enterprise organisational structures. Although O-O [Udell 1994] was later proved insufficient it inspired the development of a more complex structure, the enterprise thesaurus. The next paragraphs draws on some facts about models, enterprise structures and object orientation and shows how the enterprise thesaurus was derived.

4.3.2 Characteristics of Enterprise Models

Prior to the development of the models we need to decide where to draw the boundaries. Most enterprise models target specific processes such as supplies or sales. Enterprises are however open systems with relations to the outside world which affect their internal operations.

There are three important characteristics that should be embedded to enterprise models. These are:

- Objectives,
- Boundaries,
- Maintenance,

While the first two characteristics determine the aim and the scope of the model (i.e. a model can be used for a particular division or project within an enterprise) the third characteristic will ensure the continues use of a model. It has been reported [Whitman 1997] that most models last from the time of their creation until a major change takes place within its bounds. On a Survey conducted by Larry Whitman at Texas University it

was found that the majority of the enterprise models could not accommodate any kind of environmental change and were therefore discarded as soon as a change occurred.

4.3.2.1 Enterprise Structures

We describe enterprises in terms of roles and relationships between their internal / external components. Our definition also includes the services and resources that assist to maintain these relationships. The following figure shows the two levels of an enterprise or supply chain.

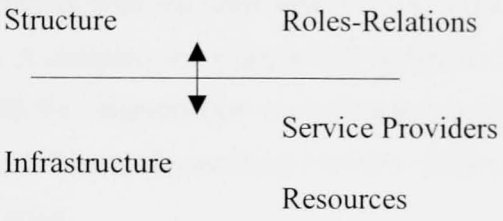

Figure 4.2 Enterprise Structure

The bottom level (infrastructure) is the basis of the top level. The two-way arrow represents the dependency between the two. As it is explained in the next paragraph the initial reaction to these facts was to describe / model the top level first (Roles - Relations) using object orientation as a vehicle.

4.3.2.2 Object Oriented Enterprise Modelling

If one tries to visualise for example a part of the automotive sector will end up with a very complex network of inter-connected nodes. The nodes in this example represent enterprises.

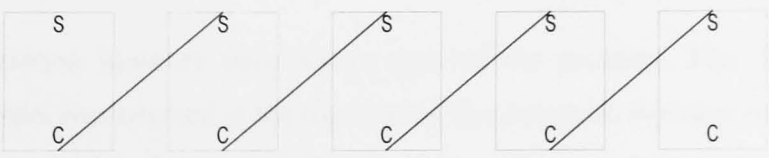

Figure 4.3 Representation of a Typical Supply Network

The structure of this network would remain the same even if we attempted to classify the nodes / enterprises according to their output due to the variety of output produced by each enterprise.

However by classifying enterprises according to the level they belong in the supply chain rather than their input and output the complex network structure of roles-relations can be transformed to a hierarchy. The top level of the classification describes the integrators whereas the following levels describe component manufacturers, component's parts manufacturers and so on and so forth. Due to the nature of supply chains level 2 objects will never be linked directly with top level objects. The connection will be done through the intermediate level. A company for example that produces and supplies leather for car seats will be linked with the company that manufactures car seats. The integrator will not be directly related. The following figure shows the new structure of a simple supply chain derived by O-O application.

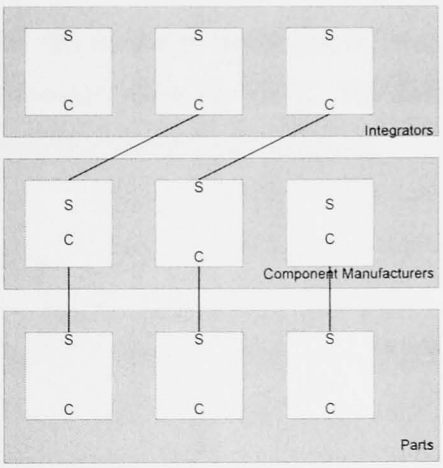

Figure 4.4 O-O representation of a supply chain

Object orientation however only solves part of the problem. This is because this particular model was targeted at the top level of the enterprise structure (roles- relations). If we attempt to add the infrastructure of the chain to the model, the hierarchical structure would collapse and the model would be transformed again to a complex network. At this

stage the O-O approach requires some sort of simplification of the model which can be either distinguishing between structure and infrastructure.

4.3.2.3 Logical Representation using Venn Diagrams

The object oriented method we devised to model the roles and relationships within enterprises solves only part of the problem. The hierarchical structure shown in figure 4.4 does not include the infrastructure of the enterprise. The infrastructure as shown in figure 4.2 can be divided in two parts. These are the service providers (telecommunications, power etc) and the logistics. We define logistics as the services that enable two objects to maintain a relationship / dependency. We use Venn diagrams to show a relationship between two enterprises and set some premises.

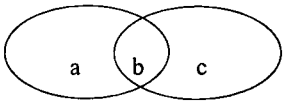

Figure 4.5 Enterprise relationships

Sets a and c represent the resources of the two objects. The intersection a∩c, or in other words set b represents the resources both objects have committed in order to maintain a relationship or conversation [Dobson 1997]. The following figure shows how service providers can be represented using Venn diagrams.

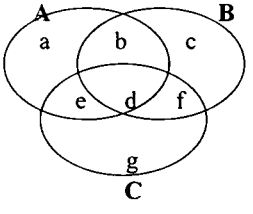

Figure 4.6 Structure / Infrastructure

Assume object C represents a service provider. Sets e and f, represents the resources used by objects A and B, in order to maintain a conversation. Set d represents the external services used to maintain the relationship of both objects. Set g describes the resources of object C: the service provider.

From figure 4.6 we can derive the following dependencies. The intersection b∩d describes a dependency (shared resources) between objects A and B. The intersection a∩e and c∩f describe the dependencies (resources used-offered) between enterprises and external service providers.

We show in figures 4.4,4.5,4.6 how relationships between enterprises are structured and how they depend upon each others operations. In practice it would be virtually impossible to describe an entire supply chain by using Venn diagrams. It is however beneficial as they offer an insight into the structures of these dependencies / relationships. Since O-O was not sufficient in representing such a structure the enterprise thesaurus method was devised. The following paragraph explains how a supply chain can be described and visualised by developing an enterprise thesaurus.

4.3.3 Enterprise Thesaurus

In information retrieval a thesaurus consists usually of a large number of subject heading (terms) which are inter-linked. Subjects are initially structured in a hierarchical way (grouping related subjects) and the links that exist between objects (subject headings) in various levels are described by predicates revealing a mesh structure. Subject headings are used to navigate from a main subject to a narrower term e.g. computing-programming. Predicates are used to link objects from one hierarchy to another by applying rules of logic. The same idea has also been adopted in developing knowledge-based systems. Knowledge is divided into frames that describe objects as well as their attributes and basic functions. The following frame shows how knowledge about a car is structured.

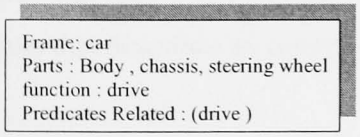

Figure 4.7 A frame representation of a car

The values of the field '*Parts*' represent the lower level of the car hierarchy. Each of them may have other frames describing their functions. The predicate 'drive' represent the true values that can be applied to the object 'car'. Since an AI system with knowledge about cars may well need to retrieve information about it is sensible to view the related memory packets (frames) as being inter-linked into some larger structure. The technique that supports such a structure is called property inheritance. The following diagram demonstrates such a structure.

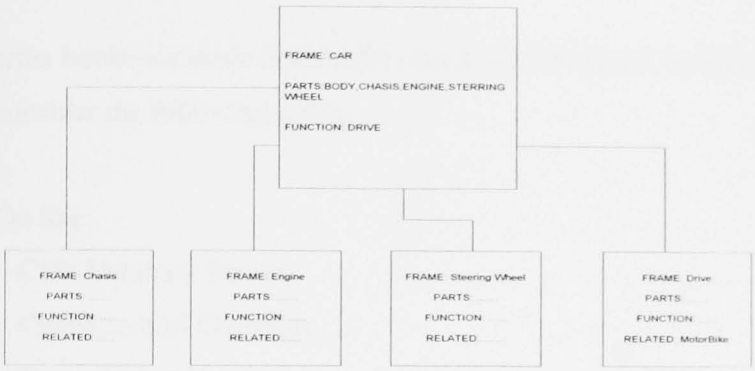

Figure 4.8. Frame Based Representation

As the diagram suggests the frame *car* related to a number of frames that describe its function, parts etc. However each of the frames could be part of some other frames structure. The frame *Drive* for example can also be part of a structure about motorbikes. The types of relationship between frames are either vertical *is-a-part-of* or horizontal i.e. peer relationships between two resources, activities etc. This particular data structure, which can also be found in thesauri enables the representation of relationships between objects that belong to different levels within a hierarchy. This representation is essential in order to represent the infrastructure of a supply chain. Drawing from previous thesaurus structures, knowledge bases, frame representations, we propose an enterprise thesaurus that describes an entire supply chain.

An enterprise thesaurus is a method for storing information about enterprises. The idea has been taken from the thesaurus structure used in information retrieval systems. Thesauri are series of terms that are linked in hierarchical form. However cross-link

between terms is also allowed and supported by some numbering system. Assume two enterprises that operate on the Internet and consider the following enterprise hierarchies:

Books On line

- Books Database Server
- Credit checking control Server
- Delivery Agent (DHL)

The terms books database and credit checking, belong to the Books On Line hierarchy. Now consider the following terms:

CD's On line

- CD's Database Server
- Credit control Checking
- Delivery Agent
  - Royal Mail
  - DHL
  - Wordmail

Again we have a hierarchy of terms called 'CD's on line' that has 2 1st level terms and 3 2nd level term. Notice that the term 'DHL' belongs to both hierarchies although in the first company is unique. Using this sort of structure in our enterprise modelling method not only we allow an initial picture of a series of interconnected agents to be developed, but also to show the various models a particular agent participates. We can use this structure to show shared resources, or shared activities between components or services within a supply chain. In the enterprise thesaurus we represent the resources that are being used to maintain a conversation as predicates. The predicates will enable relationships between agents in a hierarchy that belong to different levels. These links will be hidden from the user / modeller as the logical representation of the model's structure will remain hierarchical. The thesaurus structure is shown below.

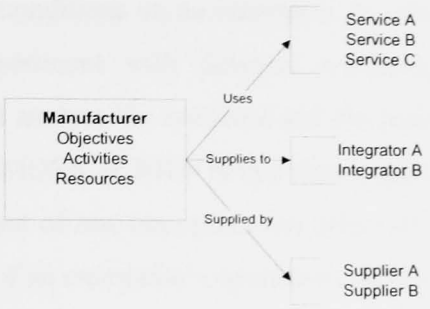

Figure 4.9 Enterprise Thesaurus

As the diagram suggests the enterprise is making use of a number of resources that were invisible in figure 4.4. Using the enterprise thesaurus we can still view the supply chain as a hierarchy representing the conversions / relationships between enterprises but we can also include the resources allocated to maintain these relationships by using predicates such as 'uses', 'used by', 'supplied to' etc., rather then direct links. Figure 4.10 shows how figure 4.4 would be represented in the thesaurus structure.

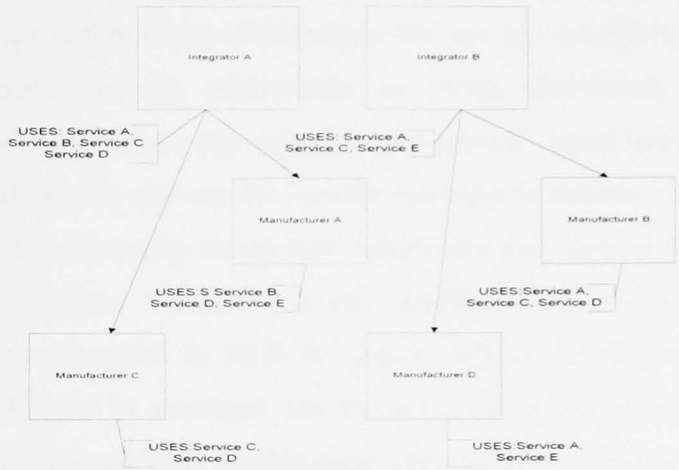

Figure 4.10 Enterprises Thesaurus

4.4 Theory of Conversations

We used a thesaurus structure to deal with the complexity of describing a network of agents. As it is mentioned above part of our aim is to define a model that will enable us to

track the impact of change in such a network. Enterprise models can help to anticipate the effect of different conditions on an enterprise. In order to minimise risk, enterprises make forecasts and experiment with fictional scenarios. They test their systems against different input and analyse the outcome and the performance. The problems with current models such as CIMOSA or IDEF is that they neglect external factors. On a supply chain the quality of output of one enterprise can affect the operations of another. How can we assess the impact of an enterprise's operations on the supply chain?

We showed on the thesaurus diagram how we link enterprises in order to declare a relationship. The theory of conversations [Dobson 1997] is posited as (among other things) a tool to analyse social, commercial and organisational roles and relationships. The term 'conversation' is used in the normative sense to describe a relationship between two agents. Each conversation is described by attributes such as: *significance, mutuality, capabilities and control.* We use the conversation theory as a vehicle to classify relationships and track the impact of changing roles and responsibilities. So, drawing from this paradigm along with each relationship we consider four variables that describe and evaluate the nature those relationships. We use the variable *significance* to indicate how the benefits of the relationship are being distributed. In the 'Theory of Conversations' *significance* can either be symmetrical (equal benefits) or asymmetrical (unequal benefits). We also use the variable *mutuality* to indicate how responsibilities are being distributed. *Mutuality* shows how responsible one enterprise is for the benefits of the another enterprise. *Capability* is the variable that describes the set of resources required by an enterprise to fulfil its responsibilities. Finally we use the *control* to indicate which agent in a network has the power to initiate or terminate a relationship. The four variables and their combinations play an important role in this method with regards to animation. The next paragraph describes them in more detail.

4.4.1 Significance & Mutuality

The following table shows all the possible combinations of the four variables and the resulting relationship types. In the first instance we will consider the values of significance and mutuality.

Conversation	Significance	Mutuality
A	On (symmetric)	On (symmetric)
B	On (symmetric)	Off (assymetric)
C	Off (assymetric)	On (symmetric)
D	Off (assymetric)	Off (assymetric)

Table 4.1 Relationship types

We use Boolean values (on – off) to evaluate the four switches of mutuality, significance, control and capability. In terms of significance and mutuality the on-off switches are used to distinguish between symmetric (benefits are equally distributed and both parties a equally responsible) and asymmetric relationships (one party benefits mostly and none of them is responsible for the benefit of the other). The above table resulted in four possible combinations of significance and mutuality. Let's take each one in turn. Conversation type A is defined in terms of symmetric significance and symmetric mutuality. It describes relationships between players on a supply chain environment. Both parties enjoy equal benefits from the relationship and both are equally responsible for the benefits of the other. This type of relationship can be observed in the automotive industry between major car integrators and component manufacturers. Many times the contractual arrangements between the two parties are such that one party is responsible for making use of 100% of the capacity of the other. The significance of this relationship is symmetrical as benefits are equally distributed. In terms of contractual arrangements the relationship is also mutual. This is because the party who initiated the relationship cannot dissolve it, (due to contractual arrangements) while the other party is not allowed to supply its capacity to other clients/agents.

Conversation type B describes supply chain relationships where although the benefits are equally distributed none of the parties is responsible for the benefit of the other. Again this type of relationship can be observed in a supply chain environment as the one described above. The different would be in the contract between the two parties where

one party selects the best possible supplier while the second is ready to supply its capacity to the best bidder. Conversation type A describes, in a sense co-operation whereas type B describes competition.

Conversation type C implies a situation where only one of the parties enjoys the benefits of the relationship although mutuality is equal. We call this type of relationship 'invalid' because it doesn't apply to supply chain relationships.

Conversation type D reveals a parent child relationship of asymmetric significance and mutuality. It implies a situation where one party is putting the effort for the benefit of the other. We also consider the variables of control and capability. The following paragraph discusses the impact of these variables on a conversation.

4.4.2 Control & Capability

The variable control indicates which of the parties in a conversation can initiate a conversation. We use the indicator *h* to point out which party has higher authority. Similarly *l* indicates parties with lower authority whereas *e* indicates equality. In a supply chain environment the indicators would show which of the parties are higher in a supply chain or have a better position in a competitive market.

Capability relates to the party's ability to access resources. Capability of an *off* value indicates that both parties are responsible for acquiring their own resources. When capability is *on* then parties may have agreed upon using or sharing resources. For example two students that work under the same department may share the same resources (the department offers) although they are independent of each other's work. Control here is equally spread as both could initiate a conversation / relationship.

Each of the conversation types is associated with a set of control and capability values. Control describes which of the parties can initiate a relationship. Generally speaking in the context of supply chains the party higher in the hierarchy (integrator) has more control over the lower levels (component manufacturers). We use the variables *h*, *l* ,e to

denote control being passed to the party higher in a hierarchy, lower or at the same level respectively. Let's see how these variables affect the conversation types A, B, C and D in the context of supply chains.

Conversation Type	Control	Capability	Relationship Type
A	L	On	Invalid
A	H	On	Exclusive co-operation
A	E	On	Teamwork
A	L	Off	Invalid
A	H	Off	Invalid
A	E	Off	Invalid
B	L	On	Invalid
B	H	On	Invalid
B	E	On	Invalid
B	L	Off	Special Unique
B	H	Off	Customer Supplier
B	E	Off	Customer Supplier (same level in the hierarchy)
D	L	On	Invalid
D	H	On	Parent Child (investment)
D	E	On	Invalid
D	L	Off	Invalid
D	H	Off	(Parent Child)
D	E	Off	Invalid

Table 4.2 Control and Capability in the Supply Chain

Note here that some relationships are classified as invalid in the context of supply chain. This does not imply that they cannot exist. The table however helps us to define the rules that will allow us to construct the set of all possible directions the model can animate. Conversation type C is invalid within this context and it has been omitted from the table.

As I mentioned earlier in the thesis this set of variables is used to evaluate the list types that describe the model. As we have already introduced the list type: *premise* that keeps track of the type of relationship an agent can be involved, we're introducing the three other list types using examples where appropriate.

4.4.3 Combining the Enterprise Thesaurus with Conversation values

We assign a set of the variables to each side of the relationships. If for example the relationships described the relationship between an integrator and a manufacturer we would use these variables to describe the relationship from each enterprise's point of view. Each of these variables is evaluated to either 1/on or 0/off. In the case studies that will follow we evaluate significance and mutuality as either symmetric or asymmetric which is translated to on or off. This is done for the sake of simplicity, so the examples can be better illustrate without going into too complex data structures. We do however acknowledge that in reality there may be a range of significance and mutuality values such as *very symmetric*, *average, almost asymmetric* etc. The representation of these values could be done using real numbers. Again for the sake of simplicity we use two distinct values 0/off and 1/on as opposed to the range of real numbers between 0 and 1. Figure 4.10 reveals the structure of the enterprise thesaurus. Consider only the relationship between Nissan and its services. The contractual arrangements between the service providers and the component manufacturers vary because the purpose, roles and responsibilities vary. We can therefore assume that the variable values between services and component manufacturers will vary too. Figure 4.11 illustrates the previous points.

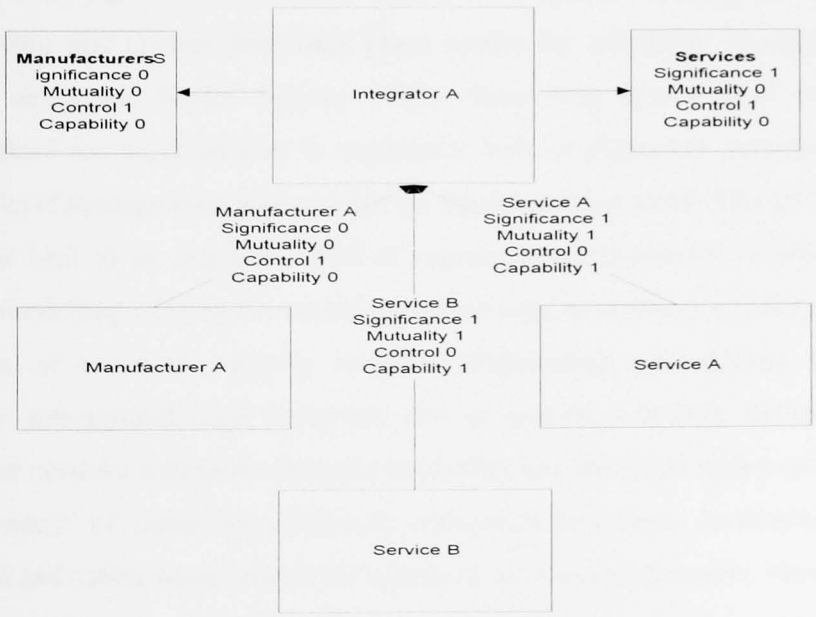

Figure 4.11 Integrator and Services

The above figure reveals that the *significance* of the relationship varies according to the type of agents involved. It also shows that Nissan (top of the chain) will maintain control in initiating relationships where the service providers hold the responsibility for providing enough resources to maintain Nissan's operations. Nissan maintains a policy (control) of the type of relationships it establishes with manufacturers and service providers. By changing the variable values we change the nature of relationship. There are however some premises that we need to consider when modelling such a structure. When for example *significance* is asymmetric then one party has to accept high mutuality (responsibility). Also if both parties accept high mutuality then both have to have the same control and capability to obtain resources. Combinations between the two set of variables reveal different relationship types. The next few paragraphs and up until the case studies the initial picture of the model built using the enterprise thesaurus and described by the conversation variable is animating using some of the AI principles mentioned earlier.

4.5 Animation

Whitman conducted a survey [Whitman 1999] assessing the re-usability of enterprise models. As he reported in his Ph.D. thesis, the majority of enterprise models fail to accommodate any change that takes place within the enterprise or externally and are therefore discarded. Seeley [Seeley 1999] claims that due to poor modelling it is commonplace for organisations to experience lack of alignment between the goals of strategic level management and realities on the operational level. This poor alignment is due to the lack of an implicit model of organisational dynamics. When we deal with dynamic modelling of complex organisations we need to consider variables such as over-estimation of practical capacity systems, fragmenting performance measures that encourage sub-optimal local decisions, and in general a holistic perspective [Seeley 1999]. The need for enterprise dynamic modelling has also been addressed [Keller 1998] in the context of modelling chemical production processes. Increasing economical, ecological and safety requirements for chemical production processes, increases demands for assessment of the predicted behaviour stage of process development [Keller 1998]. The reason most organisational models fail is that they are inflexible diagrammatic

schemes whose purpose is to model very complex dynamic systems. Similar observations have been made in the context of software engineering and information systems [Kanellis 1999]. These modelling inefficiencies can also be observed in the context of supply chains. Relationships between agents can be formed and dissolved affecting the entire chain. Although there have been some models that capture the effects of change within the boundaries of the agents that are directly affected, little has been done into assessing the indirect effects and their impact on other agents. Most times this is a result of the lack of dynamic concepts built into the model during the development stage. For the purpose of this thesis we call a model invalid when the picture it depicts is not aligned with the real situation. One has to bear in mind prior to any type of enterprise modelling that the best we can aim for is an approximation of the real situation. Unlike information systems or computing systems in general where we can model their behaviour using CASE tools and pseudo code, enterprises are dynamic systems with unforeseen behaviour. As B. Anderson [Szengheo 1999] pointed out in a recent conference [IEMC99'] one cannot model variables such as conflicts of interest, individual goals etc using type of model. Although this is an issue that could be solved with dynamic behavioural analysis of the enterprise, what is really needed is a series of dynamic models of resources, processes, structure and behaviour in order to experiment with different scenarios, make forecasts, avoid pitfalls and discover opportunities.

4.5.1 The Case of Dynamic Modelling

Before we explore dynamic modelling further we recap the following definitions. We define a conversation as a relationship between two agents and networks of agents as groups of conversations between agents within the same environment, market or industrial sector. The method we're presenting is a step into dynamic modelling of networks of agents. This modelling is not directed at the input output procedures of the chain or the logistics, there is in fact a large number of papers that address this issue. We're dealing with the relationships between agents and how their nature affects the supply chain. Modelling is being carried out by recording the movements of the model and the affects of each movement on the agents which are directly or indirectly affected. There are three dimensions incorporated into our modelling method which are: scope,

dynamics and informatics. Scope refers to the ability of the model to include service providers or agents that are not directly related with a particular environment. Dynamics refer to the ability to show the effects of new relationships been formed on all participating agents at each stage or movement of the model. Finally informatics refers to the ability of the model to show how and which type of information affects underlined agents.

4.5.2 A.I. Planning

In A.I., inference and planning of dynamic systems is directly related to the way a problem or a situation is being represented. In order to program a simple function i.e. instructing a robot to carry an item from *a* to *b*, we need to program the possible situations the robot and his environment can be in, the goal the robot has been set, the actions and sequences of actions that the robot will have to perform to achieve his goal and finally the layout of the environment. Having recorded this information in the appropriate data structures, we could produce a list of all the possible states the robot can go through in order to achieve his goal. In A.I. we call this procedure *planning*. There is however another level of complexity that deals with choosing the ideal plan that leads to the goal state in less effort and time. The aim of the approach presented in this thesis is to explore all possible states. Therefore the planning procedure could be sufficiently implemented using either breadth or depth first searches. However it is usually the case in AI research that the planners do not want to explore all states in order to reach a goal state. Evidently the planning procedure suggested here does not understand the notion of moving *towards* or *away from*. This is because the approach aims to explore rather than find a goal. In goal seeking situations however it is necessary to include in the search a mechanism that will indicate if we are moving towards or away from the goal. A common procedure for this is to endow each situation with a measure of its distance from the goal conditions. Then in order to make the move more efficient we might concentrate on those moves that reduce this distance. Relevant literature comes under the heading 'inference heuristics and search'. Another point that AI raises by its application to enterprise modelling is that of the agency. In the example earlier in the chapter the robot had agency over its own plans. In networks such as supply chains however, an agent

cannot have agency over the whole planning of the chain. Although this is true the parallelism this thesis is making is that the robot developed its plan according to a predetermined set of rules. Supply chains also operate under a series of rules and regulations that guide competition, alliances, advertising etc. Although neither of the cases determine a unique plan, we can determine all the possible ways of planning and movement. Some of the techniques are examined in the next chapter. We consider the above paradigm as appropriate for the modelling of complex enterprise networks within the same market or supply chain. The changes are represented in the form of actions that cause the state of the model to change dynamically.

4.5.3 Hypothesis

The first task we consider is the representation of the problem in a way that we can make useful inferences according to the context or state we're presented with. We need to represent the environment we're modelling along with the possible action or events that can change the state of the initial environment.

We can get a more global view of the possible changes by looking at the collection of possible situations together with the actions that can be performed in them. One can imagine the series of possible states as a hierarchy.

Each node represents a state and each arrow an action that leads to that state. The root node represents the initial situation. Thus the tree is in effect a list of all possible actions that can be performed on the model. Each node or state prior to generation is validated against a set of rules that control the options or the direction of the model. The rules help to keep the model on the right track rather than narrow the options.

4.5.4 Rules

The entire model is based on the platform of rules that describe how relationships can evolve. The rules define the boundaries or the context of the model. They prohibit a type of relationship that conflicts with the competitive environment of the supply chain. The rules are described by a series of IF...Then statements. For example if a relationship is

moving from a customer supplier state, to a co-operation, then capability is also affected. Consider the following rule

IF State(significance[on] mutuality[off]) True

AND Action(mutuality[on]) THEN

State(capability [on])

These type of rules are used in order to record and keep track of changes. They provide information as to how a state changes and what variables are affected. We name them PAD rules which stands for Preconditions-Add-Delete rules [Pratt 1994]. The PAD rules used for the case studies are incomplete. It would be impossible to provide an entire and complete set of PAD rules for this thesis. PAD rules are domain specific and they require sound knowledge of the domain that is being modelled in order to produce a complete list. Some of them can be derived from domain experts while others from past cases. The collection of these rules is likely to be a continuous process. Depending on how accurately the model can reflect a real world situation the PAD rules could be evaluated against their accuracy and completeness. PAD rules define the context of the model and therefore they remain constant throughout the modelling process.

4.5.5 PAD Rules

Some of the PAD rules are described below. The case studies presented later in the chapter are based on the following rules. The following rules illustrate only part of the entire PAD rule system.

IF State(significance[off] mutuality[off]) True

AND Action(mutuality[on]) THEN

State(invalid)

(If two agents aren't benefiting equally then conversation cannot be equal)

IF State(significance[on] mutuality[on] capability[on]) True

AND Action(capability[off]) THEN

State(mutuality[off])

(If two agents equally benefit and are equally responsible for each other then they have to share resources)

IF State(significance[on] mutuality[on] control((h) or (e)) capability[on]) True

THEN

Premise(exclusive)

(if an agent is exclusively contracted by then it cannot form any other relation)

IF State(significance[on] mutuality[on] control((h) or (e)) capability[on]) True

AND action(mutuality[off]) THEN

Premise(conversation type B)

(if an agent is being released from an exclusive relationship then it is independent and can form other conversations)

In the following paragraphs we describe how the model is represented and how we enable animation.

4.5.6 The usage of lisp type lists

We're using lists in order to describe a situation. The lists we describe are similar to lisp type programming. All lists are evaluated in terms of significance, mutuality, control and capability [Dobson 1997]. We have identified the need for 4 types of lists. These are premises, states, restrictions and actions. We examine each list type in turn.

4.5.7 Premises

Premises are list of facts that describe the nature of the enterprise. Premises denote the type of relationships that can be created as well as how relationships can evolve within the pre defined context. An integrator for example who is at the top of the hierarchy would want to maintain control over its suppliers. We record this information in lists of facts showing the values of significance, mutuality, control, and capability an enterprise is willing to accept. The lists are used to evaluate actions against them. The difference between premises and the PAD rules is that premises are used to evaluate each state the model is in whereas the PAD rules are used to evaluate the entire *tree* of options.

4.5.8 States Restrictions Actions

The evaluation process that describes a particular type of conversation is called *state*. So by *state* we imply a type of conversation that results by evaluating the four variables. Table 4.2 shows that there are a total of 8 possible states such as:

State(significance[on], mutuality[off], control[on], capability[on])

Each of these states is associated with a set of *restrictions*. As opposed to states restrictions describe the set of values that are not permitted given a state S. The reason for using restrictions is to avoid states that are potentially impossible to reach in a real situation. It could be argued that agent modelling research [Wooldridge 1996][Allen 1991] can offer a lot more flexibility in generating and calculating properties of new states using temporal modalities. This is an approach that has been adopted by model checking software such as STEP [Manna 1995]. In our approach however this could not

be the case since we do not evaluate properties against states. The rules that govern the generation of the new state change according to that state. The PAD rules prohibit the generation of states that cannot exist within the domain that is being modelled. For example a parent-child relationship cannot be formed within a supply chain domain. The premises are lists that indicate the types of relationship in which an agent can be involved and finally we use the restrictions list which were explained earlier. The major difference of our approach is that the premises of an agent change according to the state that has been reached. A condition that may have been applied to guide the generation of previous states may become invalid since premises are re-generated after every state has been reached. So this approach cannot be based on a global network of conditions that guide the entire search and enable the generation of all states. This is the reason these rules have been divided into 3 types and have been assigned different degrees of authority. We also use the predicate *action* to describe changes. At this point changes are described in terms of changing the values of the participating variables. The list below describes this concept.

State(significance[on], mutuality[off], control[e], capability[off])
Restrictions((mutuality[on] significance[off]) (control[r]))
Action(mutuality[on])

The above describes a supplier-customer conversation which can be observed in a typical supply chain. The current state tells us that the two parties are benefiting from this relationship although each has its own responsibilities towards his profits. The control switch shows that both parties can initiate a conversation and that they both have access to different resources. The restriction's list shows the values that are not permitted. Indeed if we turned the switches as they are described by the restriction list we would reach the invalid conversation defined as conversation type c. Similarly control cannot be passed only to one party. Note here that the control values h, l and e describe the position of the party in the context under which the conversation has been formed. In a supply chain environment h would mean that one party is higher in the hierarchy than the other. L implies lower position in the hierarchy whereas e stands for same level. So the

restriction on the control switch, tells us that control cannot be transferred to the other party. This implies a conversation between two parties where the first is situated higher in the supply chain hierarchy (integrator-component manufacturer).

The action statement is valid, as it does not conflict with the restriction statements. This results in a new state that is described by the following lists. Note here that the final formulation of the new state is based on the PAD rules. Further on we explain how and why.

State(significance[on], mutuality[on], control[e], capability[on])
Restrictions((mutuality[on] significance[off]) (control[r]) (control[l])

Let's examine this new state a bit further. The conversation described by this state has evolved in terms of mutuality. Both parties are responsible for each other's benefit and at the same time both are benefiting from this relationship. Control at this state remains equally distributed but since the conversation has evolved in terms of mutuality so has the capability in terms of sharing the same resources. In supply chain terms the above example described a conversation where in the first state we had an integrator – component manufacturer relationship. Both benefited from the conversation although both had control over their businesses. We can observe these conversations in the automotive for example where a tyre manufacturer may supply several integrators. In the later state the conversation evolved. The integrator accepted responsibility for the component manufacturers benefits and provided the resources the support this type of conversation. Again in the automotive we have examples where integrators (major car manufacturers such as Nissan) support component manufacturers in terms of growth (capacity). They absorb all the available capacity and prohibit the component manufacturer from supplying competitors. Another conclusion from this type of conversation is that if we examine the PAD rules we see that because this form of conversation has been formed, the premise of the component manufacturer would also change to exclusive, which means not available for other conversations. Notice here that as the examples will illustrate these rules are evaluated and generated as logical

inferences. The types of rules in the animation process are descriptive in the sense that they show how conversations can be animated. Normative type of rules however such as the PAD rules describe the context as well as the operations that can be applied for a particular type of model. Complete examples are given later in the chapter. In the next paragraph we're recapping the entire process.

4.6 Methodology

The basic concept is based on the idea to represent the changes of a supply chain as a series of states, each representing a relationship or set of relationships between two agents/enterprises. We have developed a set of PAD rules that define the possible forms the model can take in the context of supply chains. For every model or state (supply chain picture) we define a set of premises which declare the possible conversations individual agents/enterprises can be involved in. These will differ according to the level in a supply chain each object is situated. The level of the supply chain can be decided by looking at the enterprise thesaurus, which is in effect a static enterprise model itself. It describes the agents/enterprises in terms of level in the hierarchy, and resources.

Each state is associated with a set of restrictions. The list of restrictions related to each particular conversation prohibits values that will enable conversations to evolve in a way that exceeds the bounds of the supply chain. Restrictions are also validated against the PAD rules.

Actions are lists of variable values that cause the conversation to change. They are compiled first against restrictions in order to avoid situations where the conversation is invalid. Moving on we compile the state against the premises. If the state is valid then it is validated against the PAD rules that will show how the conversation will evolve.

If for example an enterprise high in the hierarchy establishes an exclusive conversation with a company, then the company's premises will change to indicate that this particular company cannot form any further relations. Similar actions are being taken when a company is being released from an exclusive conversation.

We calculate and record changes on all three lists. Premises are evaluated against the PAD rules. The variable values of each state are updated from the action lists. New restrictions are formed by the premise and the action lists in order to allow for reverse actions. The following model describes the entire process of animating the supply chain model.

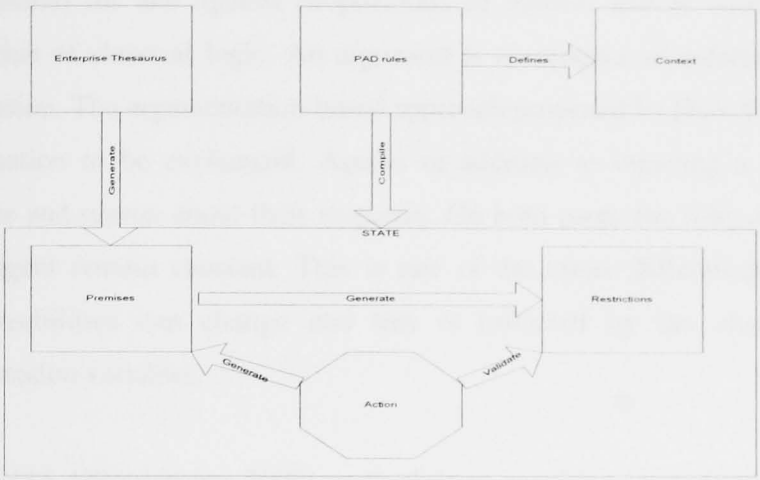

Figure 4.12 Dynamic Supply Chain Modelling

From the description of the methodology one may notice that similar concepts have been applied to enhance real time communication of agent-based systems. Relevant literature comes under the heading of negotiation and argumentation. Negotiation according to [Parsons 1998] is the process by which a group of agents come to a mutual acceptable agreement on some matter. Negotiations take place at real time and their purpose is to manage coordination between agents. By agent Parsons implies an encapsulated computer system that is situated in some environment and that is capable of autonomous action. Within this context of agent negotiations [Sierra 1999] proposed a number of domain specific negotiation protocols. Negotiation protocols are rules that govern interaction. This covers the permissible types of participants (e.g. the negotiators and any relevant third parties), the negotiation states, the events that cause negotiation states to change and the valid actions of the participants in particular states. Based on this, each agent employs a decision making model which can be viewed as the apparatus the

participants employ to act in line with the negotiation protocol in order to achieve their objectives. The model of communication between two agents is rather simple. Agent *a* proposes *x* and agent *b* either accepts and performs *x* or rejects it. The decision is based on the negotiation protocol employed.

Argumentation on the other hand is the process of constructing series of logical steps (arguments) for and against propositions of interest and as such may be seen as an extension of classical logic. An argument is a sequence of inferences leading to a true conclusion. The argumentation-based approach proposed by [Fox 1992] allows additional information to be exchanged. Agents in addition to rejecting a proposal can offer a critique and reason about their response. On both cases the roles and responsibilities of each agent remain constant. This is one of the major differences from our approach. Responsibilities can change and this is reflected by the changing values of the conversation variables.

The GAIA [Wooldridge 2000] methodology provides formalisms to allow analysts to describe arguments and negotiations between agents in terms of roles, permission, protocols, reasons and others. The following paragraphs present three case studies carried out using the method presented so far.

4.7 Case studies

In the following sections of this chapter the method will be applied in a series of models such as automotive, brokerage and health care. In general the examples will reflect the following schema.

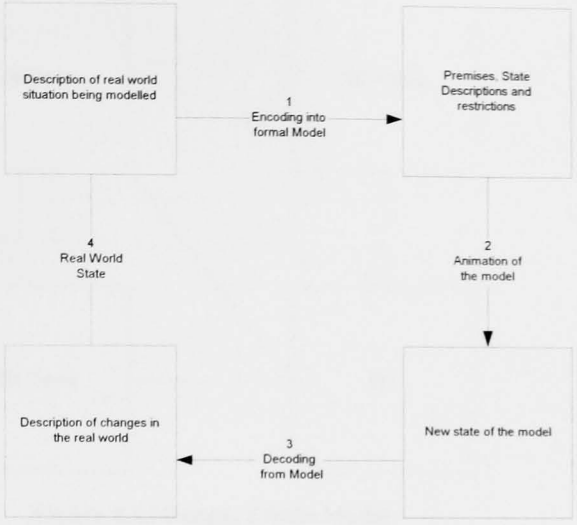

Figure 4.13 Case studies Schema

The first step involves encoding the real world situation into the model. Then the lists of premises, states and restrictions are used to calculate the new model and a model set of lists is generated.

4.7.1 Automotive

Consider the following supply chain extracted the automotive industry of the N.E. of England. The diagram show two tyre manufacturers connected to Nissan and a third object representing B.T. providing a service to all three objects. The thick arrows indicate 'uses' or 'supplied by'.

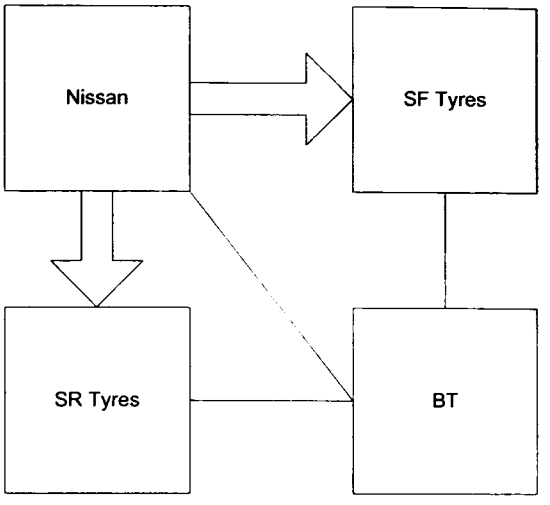

Figure 4.14 Supply Chain Model

Using the modelling method we described we could interpret the above diagram as set of lists. In the first instance we define the premises concerning each object involved in the process. Premises describe the type of conversations each object can formulate. The values of the premises themselves are derived from the enterprise thesaurus structure. The thesaurus stores relationships between objects and distinguishes between levels of hierarchy and service providers. So according to the thesaurus we could conclude the following premises.

Premises(Nissan(s[on],s[off],m[on],m[off],cn[h],cn[e],cp[off])
SR.Tyres(s[on],s[off],m[on],m[off],cn[h],cn[e],cn[l],cp[off])
SF.Tyres(s[on],s[off],m[on],m[off],cn[h],cn[e],cn[l],cp[off])
BT(s[off],m[off],cn[e],cp[off]))

State(Nissan_SR.Tyres(s[on],m[off],cn[h],cp[off])
Nissan_SF.Tyres(s[on],m[off],cn[h],cp[off])
Nissan_BT(s[on],m[off],cn[e],cp[off])
SR.Tyres_BT(s[on],m[off],cn[e],cp[off])
SF.Tyres_BT(s[on],m[off],cn[e],cp[off])
SF.Tyres_SR.Tyres(s[off],m[off],cn[e],cp[off])

Restriction(Nissan_SR.Tyres((s[off],m[off]), (cn[l])
Nissan_SF.Tyres((s[off],m[off]), (cn[l]))
Nissan_BT((s[off],m[off]), (cn[l]) (cn[h]))
SR.Tyres_BT((s[off],m[off]), (cn[l]) (cn[h]))
SF.Tyres_BT((s[off],m[off]), (cn[l]) (cn[h])))

We have used the above statements to describe the picture of this particular supply chain. We can conclude from the above statements that BT is related to all three agents although there is no direct competition between them. Both tyre manufacturers supply Nissan and also are in a competition against each other. The Nissan agent has most of the control over the supply chain. Consider the following action.

Action(Nissan_SR_Tyres(m[on]))

The above statement shows that the relationship between Nissan and SR.tyres has evolved in terms of mutuality. In supply chain terms we could say that this particular conversation has evolved from a strict supplier customer conversation to a co-operation. Before we describe the new picture we evaluate the statement against the premises of the previous state. The statement is valid and so the new state is being drawn. The following diagram shows the new relations.

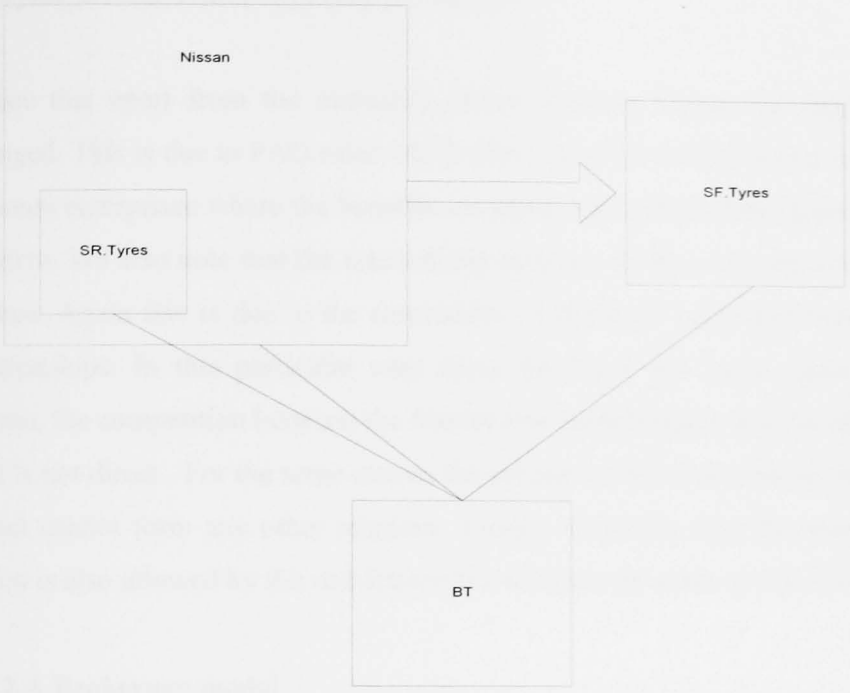

Figure 4.15 Model Movement

The new state is described with the following statements. Changes can be observed on all three lists.

Premises(Nissan(s[on],s[off],m[on],m[off],cn[h],cn[e],cp[off])
SR.Tyres(s[on],s[off],m[on],m[off],cn[h],cn[e],cn[l],cp[off])
SF.Tyres(exclusive)
BT(s[off],m[off],cn[e],cp[off]))
State(Nissan_SR.Tyres(s[on],m[on],cn[h],cp[on])
Nissan_SF.Tyres(s[on],m[off],cn[h],cp[off])
Nissan_BT(s[on],m[off],cn[e],cp[off])
SR.Tyres_BT(s[on],m[off],cn[e],cp[off])
SF.Tyres_BT(s[on],m[off],cn[e],cp[off])
)
Restriction(Nissan_SR.Tyres((s[off] m[off]) s[off], (cn[l]), (cn[e]), cp[off])
Nissan_SF.Tyres((s[off],m[off]), (cn[l]))
Nissan_BT((s[off],m[off]), (cn[l]) (cn[h]))
SR.Tyres_BT((s[off],m[off]), (cn[l]) (cn[h]))

SF.Tyres_BT((s[off],m[off]), (cn[l]) (cn[h])))

Notice that apart from the mutuality factor between Nissan the capability value has changed. This is due to PAD rules which state that if an exclusive conversation is formed between enterprises where the benefits are equal and mutual then capability must also be positive. We also note that the relationship between the two tyre manufacturers has been broken. Again this is due to the restrictions of the PAD rules that impose on exclusive relationships. In this particular case since SR.Tyres has been signed exclusively by Nissan, the competition between the former tyre manufacturer and the later has decreased or it is not direct. For the same reason the premise of SR.Tyres shows that this particular object cannot form any other relations. Finally we notice that the option to reverse the action is also allowed by the restrictions list whereas the same action is being prohibited.

4.7.2 A Brokerage model

The second model presented here is a brokerage model. The procedure for modelling the relationships between the different agents of the model is the same used for the previous example. The difference between this approach and negotiation or argumentation based approaches is that in our case we want to maintain some amount of information about the relationship types each of the agent is involved in. We want to maintain information about the relationship types an agent is involved as well as the relationship types an agent is willing to be involved in. This way we can keep track of the evolution of these relationships. All relationships have the same number of attributes based on the conversation theory. Negotiation and argumentation approaches deal with real time relationships, while the negotiation protocols determine the type of relationships that can be formed. They do not however provide a mechanism for keeping track of relationships an agent has been involved in and they also remain constant. This is something very important for our approach. Based on past relationships and their evolution we update the premises and restrictions lists. So we start the brokerage model by first developing a thesaurus. The following figure describes an initial picture of the thesaurus.

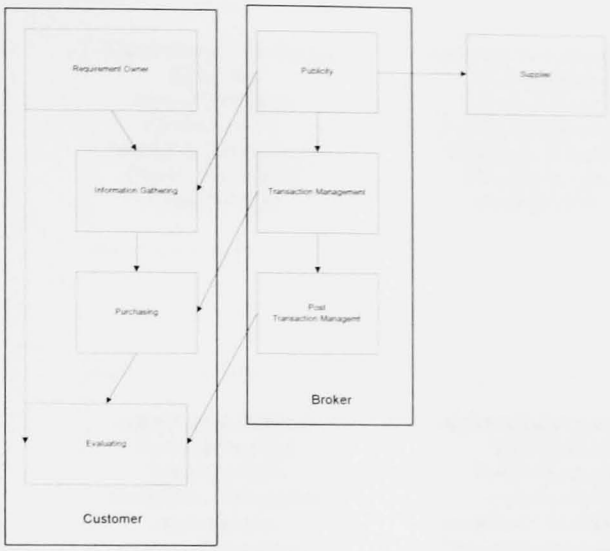

Figure 4.16 The Brokerage Model

The approach begins with a description of agents in the enterprise thesaurus. Further on we generate the States of the model based on these descriptions as well as the PAD rules. According to the PAD rules a set of premises and restrictions is also generated. This new set of lists reflects the new state of the model.

4.7.2.1 Enterprise thesaurus entries

In this initial step is to develop the first description of the model. The aim is to define the boundaries of the model as well as the relationships between agents that belong within the same agent structure or between two agents that belong to different structure. The description of all the components of this initial picture is shown in the following table.

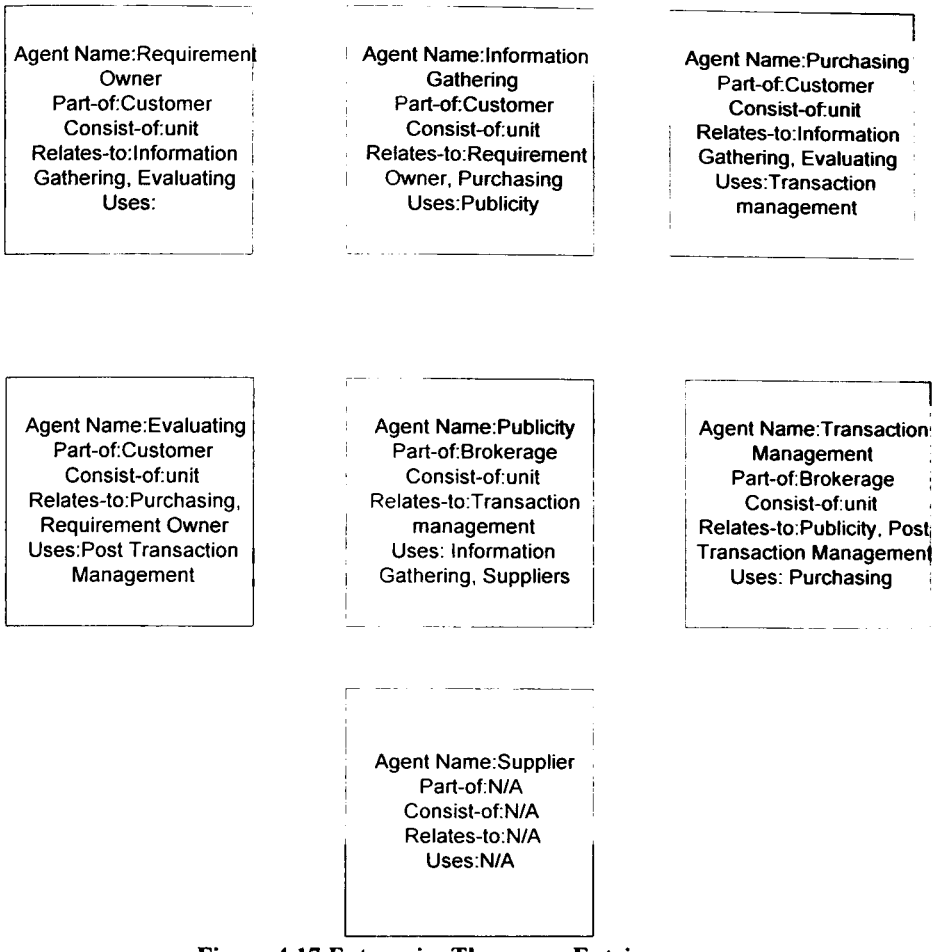

Figure 4.17 Enterprise Thesaurus Entries.

From the thesaurus one can derive the following information.

The customer agent is responsible for a number of tasks including gathering information, purchasing, as well as evaluating the process or the service provided by the brokerage agent.

The brokerage agent is a responsible for providing publicity, transaction management as well as post transaction management. The publicity or transaction management agents are connected with the sub agents of the customer structure.

Similarly the brokerage agent is connected to the supplier via the publicity agent.

There are a number of interesting conversations derived by the description of the agents which revealed the status of the current conversations and the ways the may evolve in the

future. At this point of the analysis initial schematics of the model can be developed. The thesaurus rules provide a vehicle by which the first picture of the model is being described. These rules are being used to develop the PAD rules the model will be based on.

4.7.2.2 PAD rules

We use the PAD rules to define the context of a model as well as maintain its nature and purpose of existence. The following set of rules define the relationship between customer, supplier of the brokerage agent who works as a medium between the two.

If information.gathering (invalid) then

    Requirement.owner_information.gathering(s[on],m[off],cn[e],cp[off]

End If

In other words if there is no information gathering facilities within the boundaries of the requirement agent, then the service will be provided by an external agent which of course would have impact on the type of conversation been developed by the two. Notice that mutuality remains off as well as capability, to show the two agents are autonomous.

If Evaluating.agent (invalid) then

    Post.transaction.management (invalid)

End if

The above statement indicates that if there is not an evaluating agent within the boundaries of the requirement agent then the post transaction management agent must be invalid. Notice here that in the description that to agents are up interdependent therefore if one of them does not exist then the other does not exist either.

If purchasing.agent(invalid) then

    Transaction.management (invalid)

End if

This is almost self explanatory, if there is no purchasing on the customer's behalf them there is no processing of the transaction taking place.

```
If publicity (invalid) then
        Information.gathering (invalid)
End if
```

Again these rules indicates that if no publicity has taken place the customer will not be able to gather any information.

```
If  supplier (invalid) then
        publicity (invalid)
End if
```

If there is no supplier then there is no publicity. The rules indicate that if there is no supplier in the model then the publicity agent cannot exist. In other words someone has to publicise information before they can reach the customer agent. Otherwise this type of relationship would be invalid. This is because there is a rule within the set that states if there is no publicity there is no information gathering.

```
If Requirement.Owner_Evaluating(m[off])
        Capablity[off]
End if
```

This particular PAD rule indicates that for a conversation between two agents that belong to the same structure do not have mutual responsibility towards each other and then they cannot share the same resources and therefore capability becomes negative. This particular rule is true for every conversation developed between two agents. The same rule, for example, applies to the publicity agent and the transaction agent. However only one of the rules is presented here: the rest are implied.

4.7.2.3 Generation of states

States are generated by using the rules of the thesaurus, as well as the rules in the PAD model. In this series of lists we describe the nature of conversations as well as the variable values of significance, mutuality, control and capability. The following lists describe the initial picture of the brokerage model.

State(Requirement_Information.Gathering(s[on],m[on],cn[e],cp[on]
Requirement_Evaluating(s[on],m[on],cn[e],cp[on]
Information.Gathering_Purchasing[s[on],m[on],cn[e],cp[on]
Information.Gathering_Publicity[s[on],m[off],cn[e],cp[off]
Purchasing_Evaluating(s[on],m[on],cn[e],cp[off]
Purchasing_Transaction.Management (s[on],m[off],cn[e],cp[off]
Evaluating_Post.Transaction.Management(s[on],m[off],cn[e],cp[off]

Publicity_Transaction.Management(s[on],m[on],cn[e],cp[on]
Transaction.Manag_Post.Transaction.Manag(s[on],m[on],cn[e],cp[on]
Publicity_Supplier(s[on],m[off],cn[e],cp[off]
)

Both states show the initial picture of the brokerage model. One can notice that the components of any agent's traction maintain equal control between them as well as positive culpability. The conversations between components of distinct agents maintain equally control and negative capability. Mutuality is also off on these conversations to indicate that the operation of businesses is not dependent.

4.7.2.4 Premises

The premises list show us how conversations from the initial model can evolve. Again these type of lists are developed automatically following the rules and descriptions given in the enterprise thesaurus. The following, are lists of premises regarding the brokerage model.

Premises(Requirement.Owner(s[on],m[on],m[off],cn[e],cn[h],cp[on],cp[off]
Information.Gathering(s[on],m[on],m[off],cn[e],cn[h],cp[on],cp[off]
Purchasing(s[on],m[on],m[off],cn[e],cn[h],cp[on],cp[off]
Evaluating(s[on],m[on],m[off],cn[e],cn[h],cp[on],cp[off]
Publicity(s[on],m[on],m[off],cn[e],cn[h],cp[on],cp[off]
Transaction.Management(s[on],m[on],m[off],cn[e],cn[h].cp[on],cp[off]
Post.Transaction.Manag(s[on],m[on],m[off],cn[e],cn[h],cp[on],cp[off]
Suppliers(s[off],s[on],m[on],m[off],cn[e],cn[h],cp[on],cp[on]
)

The situation described by the lists of states and premises is shown in the schematic 4.18.

4.7.2.5 Actions

Actions are lists of variable values that cause the model to animate. Consider the following lists of actions and how they impact on the current picture. Assume that the Evaluation of the information supplied by the broker is not evaluated by the customer but by some other agent. The information gathering procedure is also done by some other independent agent on behalf of the customer. Finally assume that the broker is responsible for the transaction management but not for the post transaction management.

Action(Requirement.Owner_Evaluating(s[on],m[off])
Requirement.Owner_Information.Gathering(s[on],m[off])
Transaction.Management_Post.Transaction.Management(s[on],m[off])

The above action lists show some changes to the variables of the conversations. Actions describe scenarios that the modeller or analyst may want to apply to the model. Some of them may be dealing with one agent while other with more. Complete lists of actions, for each state, can be produced by applying heuristics. The process is examined in the next chapter. This complete list of actions would result in the complete list of scenarios that can be applied given a state *s*. Their execution however is determined by the PAD rules that may prohibit a scenario from being executed. According to the methodology

developed the first thing we do is establish that the actions do not clash with the premises as well as the restrictions. Before performing the changes described by the action lists we check the PAD rules, in order to assess the impact of these changes to the overall model. From the PAD rules one can realise that the actions have caused a change, not only to have the variables described, but also to a wider set off variables. According to the thesaurus capability off implies autonomous agents. That means that although in the initial picture the evaluating, information gathering, and transaction processing agents belong to a wider structure, they have now become autonomous entities. The new picture is described by the following schematic.

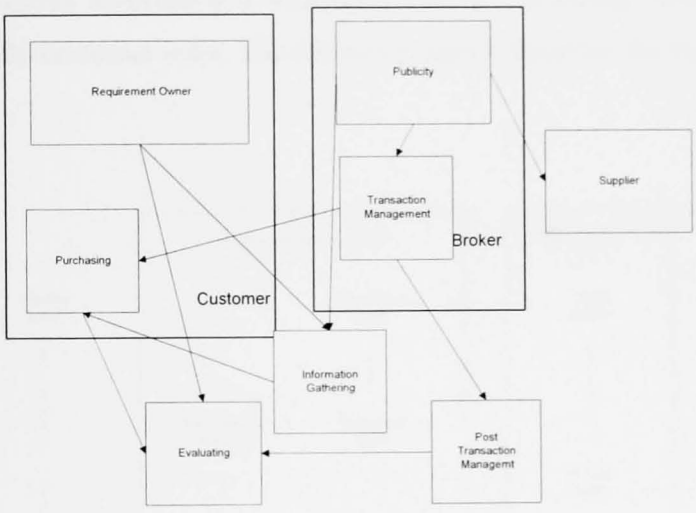

Figure 4.18 Brokerage model new state

The following lists describe the new version of the model as it is depicted in 4.18.

State(Requirement_Information.Gathering(s[on],m[off],cn[e],cp[ff]
Requirement_Evaluating(s[on],m[off],cn[e],cp[off]
Information.Gathering_Purchasing[s[on],m[on],cn[e],cp[off]
Information.Gathering_Publicity[s[on],m[off],cn[e],cp[off]
Purchasing_Evaluating(s[on],m[off],cn[e],cp[off]
Purchasing_Transaction.Management (s[on],m[off],cn[e],cp[off]
Evaluating_Post.Transaction.Management(s[on],m[off],cn[e],cp[off]

Publicity_Transaction.Management(s[on],m[on],cn[e],cp[on]
Transaction.Manag_Post.Transaction.Manag(s[on],m[off],cn[e],cp[off]

This series of lists shows the new state of the model described in the schematics.

4.7.3 Health Service

The following paragraph describes the conversations between agents in the healthcare service. The difference between this type of model and the product supply chain is that the roles of agents involved in a healthcare system can change with a higher frequency than the supply-customer roles. The following figure describes the healthcare model.

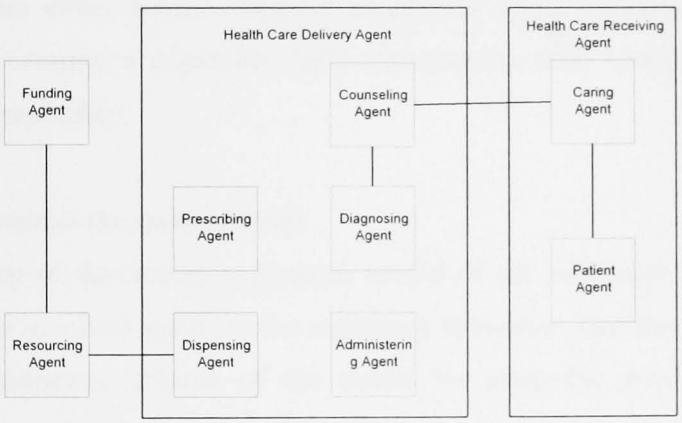

Figure 4.19 Healthcare model

The figure shows three types of agents dominating the healthcare model. The funding and resourcing agent can be thought of as the government body responsible for raising funds and supplying the delivery agent. The delivery agent supplies a series of services (as described by the boxes) to the receiving agent (the patient). Let's describe the above relationships as they appear in the figure using the method described in the thesis.

The relations between the three distinct agents are described first using lists of states.

State(Funding.Agent_Delivery.Agent(s[off],m[off],cn[h],cp[off]
Delivery.Agent_Receiving.Agent(s[off],m[off],cn[h],cp[off])
)

The state description reveals a parent-child as described in the methodology. It also reveals that the funding body (government) invests in the development of an autonomous service, hence capability is off. Control is passed on to the funding agent as indicated by the control [h] variable. The same conversation takes place between the delivery agent and the receiving agent. Again a parent-child conversation is revealed with control maintained by the delivery agent. This type of state also indicates one-to-many conversation between agents that have control over and agents on the receiving end of the conversation. The same type of relations has been found in supply chain where manufacturers either buyout supplier enterprises hence capability [on] or sign them exclusively creating a capability [off] conversation with symmetric significance and asymmetric mutuality.

4.7.3.1 Enterprise thesaurus entries

The first step of developing a dynamic model of the healthcare system is to enter the details of the involved agents in the enterprise thesaurus. The thesaurus will generate the initial diagrammatic schema of the model by using the details of its entries. The predicates part-of and consist-of define the boundaries of the model. The predicate uses define relationships between objects that do not belong in the same structure. The object 'resourcing' for example is part of the government structure although it 'uses' the dispensing agent'. The diagram below shows a logical representation of the entries in the enterprise thesaurus.

Figure 4.20 Thesaurus entries for the healthcare model

4.7.3.2 Premises

According to the methodology the first step of is to establish the list of premises. Premises are list of variable values that describe the context of the model. The general picture of the healthcare system consists of three agents. The premise lists will show the type of conversations these agents can be evolved in. Defining premises is a first step into building a dynamic concept prior any modelling. The premises for the healthcare service are as follows:

```
Premises(Funding.Agent(s[off],m[off],cn[h],cp[off]
        Delivery.Agent(s[off],m[on],m[off],cn[on],cn[off],cp[off]
        Receiving.Agent(s[on],s[off],m[on],m[off],cn[on],cn[off],cp[on],cp[off]
    )
```

The above premises describe the conversation types agents are willing to be involved into. The list also reveals that the strongest or the most powerful body in this model is the funding agent. The list shows the funding agent is the less flexible with regard to the

conversations it is willing to accept. The delivery agent is more flexible but with some restrictions whereas the receiving agent is the most flexible agent. This point really demonstrates one of the advantages of the approach. This type of flexibility cannot be demonstrated or exercised using only diagrammatic schemas. Notice that premises are lists that are being changed dynamically unlike PAD rules which remain static throughout the process giving a sense of boundaries to the model.

4.7.3.3 PAD rules

The following rules describe the conversations that can be formed within the context of a healthcare service. The rules also show which agents have to be present in order to establish the model. The presence of a diagnosing agent or a prescribing agent for example is essential, whereas the receiving agent may or may not have a caring agent.

```
If Counselling.Agent_Patient(s[off],m[off],cn[h],cp[off]) TRUE then
        Counselling.Agent_Caring.Agent(invalid)
        Caring.agent_Patient(invalid)
End If

If Caring.Agent(invalid) TRUE then
        Counselling.Agent_Patient(s[off],m[off],cn[h],cp[off])
End If

If Counselling.Agent(invalid) TRUE then
        Administering.Agent.Patient(s[off],m[off],cn[h],cp[off])
End If

If Prescribing.Agent(invalid) TRUE then
        Diagnosing.Agent(invalid) False
        Administering.Agent_Dispensing.Agent(s[on],m[on],[cn[e],cp[on])
End If
```

If Diagnosing.Agent(invalid) TRUE then

Prescribing.Agent(invalid) False

End If

If Caring.Agent(invalid) TRUE then

Patient.Agent(invalid) False

End If

4.7.3.4 States

The following paragraphs describe the state of conversations between agents within the three main agent bodies.

Funding Body

State(Funding.Agent_Resourcing.Agent(s[off],m[off],cn[h],cp[off])

In this type of relation, the resourcing agent works under the supervision of the funding agent and delivers a service to the healthcare delivery agent. The funding agent is responsible for delivering the funds through the resourcing agent. The resourcing agent is responsible to deliver the funds. Control is maintained by the funding body. However both bodies are autonomous hence capability [off]

Healthcare Delivery

State(Diagnosing.Agent_Prescribing.Agent(s[on],m[off],cn[h],cp[on]

Diagnosing.Agent_Counselling.Agent[s[off],m[off],cn[e],cp[on]

Diagnosing.Agent_Dispensing.Agent(idle)

Diagnosing.Agent_Administering.Agent(idle)

Prescribing.Agent_Dispensing.Agent(s[on],m[on],cn[h],cp[on]

Prescribing.Agent_Councelling.Agent(idle)

Prescribing.Agent_Administering.Agent(idle)

Dispensing.Agent_Administering_Agent(s[on],m[off],cn[h],cp[on])

Dispensing.Agent_Counselling.Agent(idle)

Counselling.Agent_Caring.Agent(s[off],m[off],cn[h],cp[off]

)

The above set of conversations describe the role of each agent type within the healthcare delivery service. Conversations that do not exist are declares as idle. Capability is [on] because they all work as part of the same agent. Capability [on] in conjunction with control [h] indicates teamwork or co-operation. The counselling agent is related to the caring agent with the same conversation attributes as the healthcare delivery agent is related to the healthcare receiving agent. This is true as the counselling agent and the caring agent are the bodies through which the delivery and receiving agents are connected.

Healthcare Receiving
State(Caring.Agent_Patient(s[off],m[off],cn[on],cp[on])

In the receiving end the caring agent has established a parent-child conversation with the patient. Control is maintained by the caring agent as it is directly connected to the healthcare delivery service. Capability is on as both are part of the same body (receiving agent).

4.7.3.5 Restrictions

Each of these states describes the 'as-is' situation of the model. As the methodology suggests prior to any animation of the model a set of restriction lists needs to be established. The restriction lists show the variable values that are not permitted within the context of the model. The first listing of restrictions is associated with the general picture of the model described by the following state:

State(Funding.Agent_Delivery.Agent(s[off],m[off],cn[h],cp[off]
Delivery.Agent_Receiving.Agent(s[off],m[off],cn[h],cp[off])
)

The restrictions for this state within the context of the healthcare service are the following:

Restrictions(Funding.Agent_Delivery.Agent(s[on],m[on],cn[l],cn[e])
Delivery.Agent_Receiving.Agent(s[on],m[on],cn[l],cn[e],cp[on])

4.7.3.6 Funding Agent

For the conversation type:

State(Funding.Agent_Resourcing.Agent(s[off],m[off],cn[h],cp[off])

The restrictions that apply are:

Restrictions(Funding.Agent_Resource.Agent(s[on],m[on],cn[l],cn[e],[on])

The restriction list shows that the funding agent stays in control of the conversation. Regarding the healthcare delivery agent the following restrictions are being defined for each of the individual conversations.

The states are:

State(Diagnosing.Agent_Prescribing.Agent(s[on],m[off],cn[h],cp[on]
Diagnosing.Agent_Counselling.Agent[s[off],m[off],cn[e],cp[on]
Diagnosing.Agent_Dispensing.Agent(idle)
Diagnosing.Agent_Administering.Agent(idle)
Prescribing.Agent_Dispensing.Agent(s[on],m[on],cn[h],cp[on]
Prescribing.Agent_Councelling.Agent(idle)
Prescribing.Agent_Administering.Agent(idle)
Dispensing.Agent_Administering_Agent(s[on],m[off],cn[h],cp[on])
Dispensing.Agent_Counselling.Agent(idle)
Counselling.Agent_Caring.Agent(s[off],m[off],cn[h],cp[off]

)

Restrictions(Diagnosing.Agent_Prescribing.Agent(s[off],cn[l],cn[h])

Diagnosing.Agent_Counselling.Agent[m[off],cn[h],cn[l])

Prescribing.Agent_Dispensing.Agent(cn[l],cn[h])

Dispensing.Agent_Administering_Agent(cn[h],cn[l])

Counselling.Agent_Caring.Agent(s[on],m[on],cn[l]

)

The above restrictions show the actions that cannot be applied to the above conversations. Some of them can be represented by one agent. Dispensing and administering duties for example, can be performed by the same body. The patient may not always be related with a counselling agent is another example. Finally the restrictions regarding caring agents and patient agents are as follows:

The current state is:

State(Caring.Agent_Patient(s[off],m[off],cn[on],cp[on])

The restrictions are:

Restrictions(Caring.Agent_Patient(s[on],m[on])

The current restrictions forbid the caring agent to establish a conversation different to the parent-child conversation defined in the methodology.

4.7.3.7 The picture so far and actions

So far the picture of the health care service as it is being portrayed in figure can be translated into this series of lists.

Premises(Funding.Agent(s[off],m[off],cn[h],cp[off]

Delivery.Agent(s[off],m[on],m[off],cn[on],cn[off],cp[off]

Receiving.Agent(s[on],s[off],m[on],m[off],cn[on],cn[off],cp[on].cp[off]

)

State(Funding.Agent_Resourcing.Agent(s[off],m[off],cn[h],cp[off])

State(Diagnosing.Agent_Prescribing.Agent(s[on],m[off],cn[h],cp[on]
Diagnosing.Agent_Counselling.Agent[s[off],m[off],cn[e],cp[on]
Diagnosing.Agent_Dispensing.Agent(idle)
Diagnosing.Agent_Administering.Agent(idle)
Prescribing.Agent_Dispensing.Agent(s[on],m[on],cn[h],cp[on]
Prescribing.Agent_Councelling.Agent(idle)
Prescribing.Agent_Administering.Agent(idle)
Dispensing.Agent_Administering_Agent(s[on],m[off],cn[h],cp[on])
Dispensing.Agent_Counselling.Agent(idle)
Counselling.Agent_Caring.Agent(s[off],m[off],cn[h],cp[off]
)

State(Caring.Agent_Patient(s[off],m[off],cn[on],cp[on]
)
Restrictions(Funding.Agent_Resource.Agent(s[on],m[on],cn[l],cn[e],[on])
)
Restrictions(Diagnosing.Agent_Prescribing.Agent(s[off],cn[l],cn[h])
Diagnosing.Agent_Counselling.Agent[m[off],cn[h],cn[l])
Prescribing.Agent_Dispensing.Agent(cn[l],cn[h])
Dispensing.Agent_Administering_Agent(cn[h],cn[l])
Counselling.Agent_Caring.Agent(s[on],m[on],cn[l]
)
Restrictions(Caring.Agent_Patient(s[on],m[on])

)
Consider the following action and the impact it has on the model.

Action(Caring.Agent[invalid])

The action indicates that the caring agent has left the model. The caring agent is part of the receiving agent so this is area where the changes will occur. In order to find the impact on the model, and the changes this departure has brought, the action is being validated against the restrictions lists. Later on the premises list is updated as well as the new state. The new state describes how the diagrammatic scheme of the model has evolved. The restrictions lists show that the action is valid so the model assesses the impact as follows:

If the action forms part of the PAD rules then the new model is formed following the corresponding If...Then statement. Premises are also being updated as well as restrictions. If the action isn't part of the PAD rules then the variable value is updated and recorded in the premises. Restriction lists are updated using the premises lists. The following figure shows how the new model is being formulated.

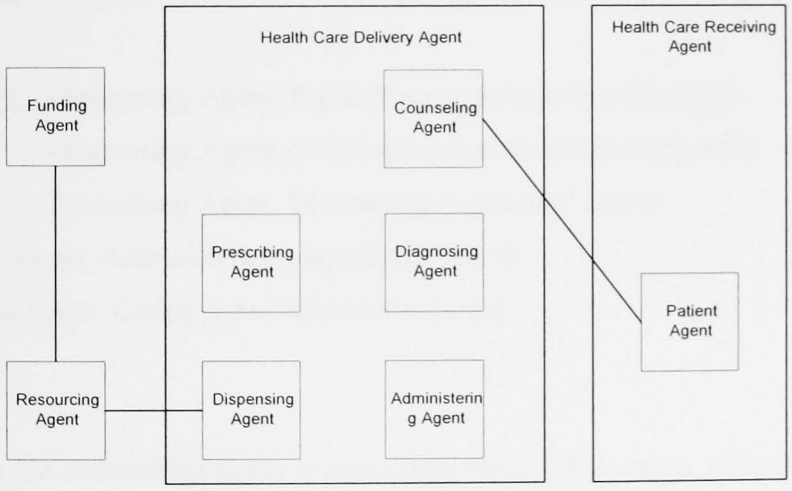

Figure 4.21 Health Care animated new state

The new state is described by the following statements.

Premises(Funding.Agent(s[off],m[off],cn[h],cp[off]
Delivery.Agent(s[off],m[on],m[off],cn[on],cn[off],cp[off]
Receiving.Agent(s[on],s[off],m[on],m[off],cn[on],cn[off],cp[on],cp[off]
)

State(Funding.Agent_Resourcing.Agent(s[off],m[off],cn[h],cp[off]

State(Diagnosing.Agent_Prescribing.Agent(s[on],m[off],cn[h],cp[on]
Diagnosing.Agent_Counselling.Agent[s[off],m[off],cn[e],cp[on]
Diagnosing.Agent_Dispensing.Agent(idle)
Diagnosing.Agent_Administering.Agent(idle)
Prescribing.Agent_Dispensing.Agent(s[on],m[on],cn[h],cp[on]
Prescribing.Agent_Councelling.Agent(idle)
Prescribing.Agent_Administering.Agent(idle)
Dispensing.Agent_Administering_Agent(s[on],m[off],cn[h],cp[on])
Dispensing.Agent_Counselling.Agent(idle)
Counselling.Agent_Patient(s[off],m[off],cn[h],cp[off]

)
Restrictions(Funding.Agent_Resource.Agent(s[on],m[on],cn[l],cn[e],[on])
)
Restrictions(Diagnosing.Agent_Prescribing.Agent(s[off],cn[l],cn[h])
Diagnosing.Agent_Counselling.Agent[m[off],cn[h],cn[l])
Prescribing.Agent_Dispensing.Agent(cn[l],cn[h])
Dispensing.Agent_Administering_Agent(cn[h],cn[l])
Counselling.Agent_Caring.Agent(s[on],m[on],cn[l]
)

Notice that the counselling agent is now connected to the patient without going via the caring agent. This new conversation is derived from the PAD rule which states that is no caring agent is present then the delivery agent is connected to the receiving agent the patient agent. Let's consider another action.

Action(Prescribing.Agent(invalid)
Counselling.Agent(invalid)

Again the above actions are validated against the restriction list and are find valid. The PAD rules describe the state new state of the model and the new conversations that arise since the departure of the prescribing and counselling agents. The PAD rules state that:

If Counselling.Agent(invalid) TRUE then

Administering.Agent.Patient(s[off],m[off],cn[h],cp[off])

End If

If Prescribing.Agent(invalid) TRUE then

Diagnosing.Agent(invalid) False

Administering.Agent_Dispensing.Agent(s[on],m[on],[cn[e],cp[on])

End If

The first rule states that if the counselling agent is missing the administering agent has to be related with the patient agent. The second rule states that if the prescribing agent does not exist in the model then the diagnosing agent must be present and be related to the administering agent. The diagrammatic scheme of the model is shown below.

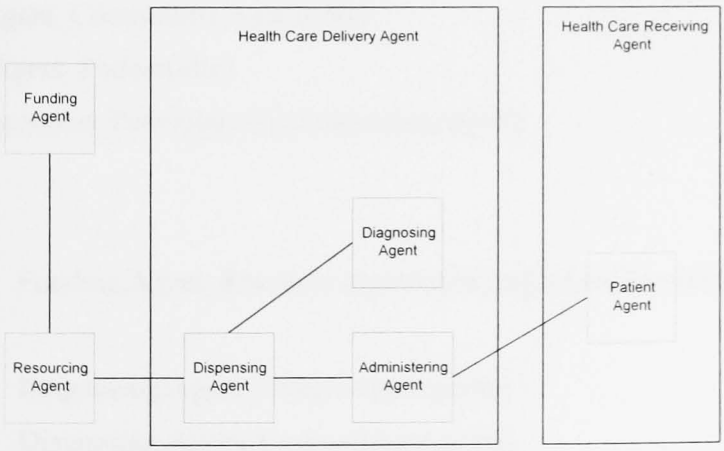

Figure 4.22 Model Transformation

In new schematic the service of the delivering agent has been passed on to the administering agent. The conversations carries the same properties as the one between the counselling agent and the patient agent or the counselling agent and the caring agent as shown in the previous figures. The new set of states are described as:

Premises(Funding.Agent(s[off],m[off],cn[h],cp[off]
Delivery.Agent(s[off],m[on],m[off],cn[on],cn[off],cp[off]
Receiving.Agent(s[on],s[off],m[on],m[off],cn[on],cn[off],cp[on],cp[off]
)
State(Funding.Agent_Resourcing.Agent(s[off],m[off],cn[h],cp[off]

State(
Diagnosing.Agent_Counselling.Agent[idle]
Diagnosing.Agent_Dispensing.Agent(s[on],m[on],cn[h],cp[on]
Diagnosing.Agent_Administering.Agent(idle)
Prescribing.Agent_Dispensing.Agent(idle
Prescribing.Agent_Councelling.Agent(idle)
Prescribing.Agent_Administering.Agent(idle)
Dispensing.Agent_Administering_Agent(s[on],m[off],cn[h],cp[on])
Dispensing.Agent_Counselling.Agent(idle)
Counselling.Agent_Patient(idle)
Administering.Agent_Patient(s[off],m[on],cn[e],cp[off]

)
Restrictions(Funding.Agent_Resource.Agent(s[on],m[on],cn[l],cn[e],[on])
)
Restrictions(Diagnosing.Agent_Prescribing.Agent(i)
Diagnosing.Agent_Counselling.Agent()
Prescribing.Agent_Dispensing.Agent()
Dispensing.Agent_Administering_Agent(cn[h],cn[l])
Counselling.Agent_Caring.Agent()

Administering.Agent_Patient(s[on],m[on])

)

The above states explain the new state of the model. Notice that the conversations where the departed agents where involved have now been turned to idle. Their content in terms of variable values has been transferred to the newly formed conversations. Another advantage of this particular type of description is that the allocation and reallocation of responsibilities in the model becomes visible. In the last figure one can see that the diagnosing agent is responsible for reaching the dispensing agent and the administering agent carries the responsibilities of the counselling agent who departed.

4.8 Conclusions

In this chapter we addressed the need for treating enterprise models as open systems. We also pointed out the need for a modelling method that is descriptive and concise enough to accommodate the complex nature of networks of interconnected agents. Finally we made a case for constructing dynamic concepts in to the modelling method so that it allows model to animate. We devised a modelling method that is based on the enterprise thesaurus and the AI planning literature. The enterprise thesaurus can be thought of as a database structure for modelling the initial stage of the real world situation. It allows identification of peer to peer relationships as well as is-part-of relationships. The cross links that are supported show shared resources, activities, agents etc. Artificial Intelligence planning was used as the basis for developing dynamic concepts into this enterprise modelling method. The result was a lisp type approach to modelling real world situations by transforming diagrams and schematics into sets of lists. The various types of lists that we used described the properties of the model in terms of the nature of conversations as well as how these can be animated. Premises, restrictions and the PAD rules provide the evaluation mechanism by which each state of the model is being evaluated. The method was tested and evaluated against three models; a simple supply chain model, a broker and the healthcare model. After examining the results of each exercise we identified a number of commonalities regarding the conversations values

between certain types of agents. These were later translated into rules for developing enterprise thesauri and PAD rules.

Chapter 5 Knowledge Representation and Inference

5.0 Introduction

Chapter 4 explains in detail a way of developing and animating enterprise models. The schema used for representing this new method was a four steps approach that involved development of the initial picture from the enterprise thesaurus and encoding of the model in lisp type lists. The third step involves animation of model by processing the lists (given an action *a*) and finally decoding of the model into a diagrammatic form. The assumption that was made in those case studies was that actions would request for changes that could be achieved by moving the model just once. The question that arises from these cases studies is how can the model be animated in order to reach a goal state (expressed in action *a*) when the goal state requires the model to be moved more than just one time.

In this chapter we discuss heuristics and heuristic searches in order to show how goals (expressed in action lists) can be found when we need to animate the model more than once. Some of the ideas can be found in model checking software although our approach differs considerably. Model checkers, such as SPIN [Holzmann 1997] and STEP [Manna 1995], are software packages that support the design and verification of asynchronous process systems. SPIN verification models focus on the correctness of process interactions, and they attempt to abstract as much as possible from internal sequential computations. To perform verification SPIN takes a correctness claim that is specified as a temporal logic formula. The states of the system are then expanded and the formula is checked. If a state satisfies the conditions of the formula then the analyst can take appropriate action. It is undoubtedly a very useful technique when one needs to identify mismatches, deadlocks or other similar issues. The procedure for generating states and the mechanisms for calculating the distance from the goal are hidden. It is up to the model checker to employ the appropriate heuristics for searching. The goal is specified in the formula and the analyst is interested in the evaluation of that formula. Our approach differs considerably from model checkers since we are not seeking a goal and we do not

evaluate propositions as such. The primary objective is to reveal and explore possible combination of roles and responsibilities. The approach aims to maintain track of the model's movement and this makes the use of a particular heuristic algorithm important. An analyst may wish to expand a particular nodes in all its possible states as opposed to expand all nodes one step at a time. This is why decisions such as breadth first as opposed to depth first search are necessary. Notice that in this chapter some of the heuristic or knowledge representation methods presented in the examples may not be directly related to enterprise modelling. They are however presented as a means for further research that could assist or eventually become related with the field of enterprise modelling. The ideas that have been adopted by this approach are clearly pointed out.

In chapter 4 we showed how rules could help in animating the model. We explained that these rules define the domain of the model and determine the ways the model can move as well as the consequences of moving a model. Although in chapter 4 these rules were presented as a series of *if...Then* conditions in reality it is a lot more complex. However literature on knowledge engineering and knowledge representation has taught us ways of tackling the problem of representing complex data structures. Later on in the chapter we also discuss how knowledge bases can be constructed so that they can support the complex nature of enterprise modelling.

5.1 Multiple animations

The aim of this section is to examine ways of finding business roles and responsibilities where the enterprise model needs to be animated more than one times. It can be the case where an action calls for changes that require more than one movement of the model in order to reach the goal i.e. execute the action. One of the widely used methods in AI literature is to create a tree of possibilities covering all the possible movements of a given model. The outcome in general terms is a tree of nodes, each one representing a new state. Each node on the tree will either move closer to the desired goal i.e. action list or away. One node will eventually represent the goal-state as it is stated in action *a*. It should be stressed here that the purpose of this exercise within the context of the enterprise model may not always be to reach a goal-state. In fact the goal state may not

exist may not be feasible to be reached due to restrictions applied by PAD rules. It could be alternatively used to explore possible states in terms of combinations of roles and relations. So in the following paragraphs we define a goal as a combination of roles expressed in an action list or as a series of combination that serve the purpose of exploration. We will examine ways of generating possibility trees, assessing each new state against the goal-state and finding the optimum plans. By optimum plans we mean, determining the least number of movements in achieving the desired state. Progressing towards the goal is achieved by making inferences based on either a set of beliefs or heuristics. The following paragraphs examine these concepts.

5.1.1 Inferences & Definitions

A primary aspect of inference is belief versus heuristics. Beliefs are propositions that are known to be true. Heuristics are mechanisms that we employ to infer the value of a proposition. What makes either machine or man good at inference is not just a matter of beliefs but the strategies used to deploy these beliefs and rules of inference effectively. Heuristics can be differentiated from beliefs. Beliefs can be divided into true or false, whereas this is not possible for heuristics that are spoken of as effective or ineffective. Consider the following sentence:

If P conclusion is inferred on the basis of a belief Q, which is now found to be false it is reasonable for not just Q but P to be given up too. However if P conclusion was inferred using a round about proof and later changed by a problem solving method (heuristics) then this case it is not rational to give up the conclusion P as inference is still acceptable.

Consider another example: A & B consider the same question from different assumptions. If A tells B a conclusion P with reasons, it is not reasonable for B to take on conclusion P because they had different premises. However, if A & B consider same question have same assumptions – using different heuristics, and A tells B conclusion P there is no reason for B not to accept conclusion P.

A basic overview of characteristics which differentiates beliefs and heuristics, is that people show flexibility with heuristics which they do not show with regard to their beliefs. If a reasoner's beliefs and heuristics work together to produce inferences, his beliefs provide access to inferences which otherwise would not be possible. To characterise a reasoner as having beliefs and heuristics and being able to show flexibility with one and not the other implies complexity. However these characteristics cannot be described as belonging to any mechanical device, especially as machines would not be able to portray flexibility and thus it follows that such devices are incapable of making inferences, as they lack the requisite structure to allow characterisation in terms of interacting beliefs and heuristics. It is true that it is few, if any, AI systems which can display as much flexibility as that which is expected from humans and which characterise the heuristic nature of inference. However the potentiality for flexibility is there, as the system's architecture is based on separation of knowledge-base and heuristics, and whose output can be heuristically guided through the search-tree. If such a system was given the added quality of openness that is important for inference, which is primary for AI; such architectures hold the possibility of supporting heuristic flexibility, which is a characteristic of the inferential human behaviour.

In AI this division into interacting components – beliefs and heuristics – is basically modelled by a search-based architecture, in which the 'knowledge base' (beliefs) is distinguished from the heuristics. The knowledge base, together with the system's rules of inference, defines a search tree (PAD rules and premises lists). The separate heuristic component then directs the system's actual inferences, possibly restricting the inferences that it will in practice make. In the approach outlined in chapter 4 the set of beliefs is defined by the PAD rules that are in a sense the vehicle which guides the animation of the model. Although in chapter 4 these beliefs are presented as a series of conditional statements, later on in this chapter we go into detail about how these beliefs can be constructed in knowledge based systems. So the questions that arise after these definitions is 'how do we search' and more importantly 'how do we find a plan that leads to the goal state'.

5.2 Searching for best plans

This following section looks at searching methods and the advantages and limitations of each method. The first type of search methods to be discussed are those of depth-first, bounded depth-search and breadth-search. When we search we need to consider the order in which we develop the nodes, what we encounter in the search space as well as the direction. It is probable to think we have to be systematic about this, and if so, there are obvious principles which can be used, and which produce different searches with different characteristics. First we could expand fully nodes in the order in which they are encountered in which case we will perform a breadth first search, or we could develop the most recently encountered node, in which case we will perform at a depth first search. To put in into perspective consider the example of the health service as it was presented in chapter 4.

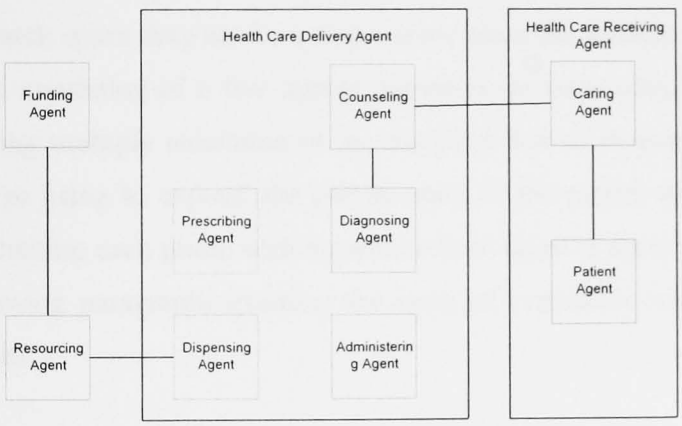

Figure 5.1.Health service

Adapting breadth first search movement into this model would require to fully expand every node, starting with the funding agent, and exploring all the different roles and relationships that it can be involved in. This would create a branch of possible movements for the funding agent. On the other hand depth first search would require the movement of every node to be considered in parallel. Every agent would be expanded one level before going to the next level of movement for all of them. The depth first searches are similar to what we do when we look around a foreign place. We must explore each node by physically moving to that node and we have no choice but to continue our search by developing a next node. With this type of search we only have one

node open at a time and this has the advantage of being economical on memory. However there are two major problems with this search methods. First of all what we meet may set off on entirely the wrong track and may not find the goal state at all or even worse the path may not terminate and the search would continue until the resources on the machine run-out. This problem can be limited, by limiting the depth to which we will explore any given path and backtrack when that bound has been reached. It is necessary to make sure that the solution exists within the depth bound.

An alternative search method is that of breadth-first search, and this explores the search space to a given depth before expanding any nodes to greater depth. Thus we're avoiding the problems associated with depth first search, because we are assured of finding a solution if one exists. However the disadvantage is that a lot of memory will be needed because numerous nodes will be open simultaneously. In the method we presented memory and search space may not be a major issue since the models we examined were relatively small, consisting of a few agents. However an issue which may be of major importance during multiple animation of the model is that of evaluation. Regardless of the method we're using to expand the current state of the model we're faced with the problem of evaluating each phase with regards to how close or far it gets from the goal-state. The following paragraphs examine the issue of evaluation within the context of heuristic searches.

5.2.1 Heuristic or ordered search method

Heuristics is the method by which the search space is evaluated and expanded. In the previous example regardless of the method we use to expand our model, we use PAD rules for evaluating each stage. Some of the relationships or new roles will be prohibited and therefore omitted. If each stage of the model was also evaluated against some action that described the goal state, then heuristics would eventually create a path to that stage. The following sections explain searching methods as well as search patterns. Neither breadth-first or depth-first search methods are the ideal answer. In reality search space is a very large, and neither method can cope with the complexity. Playing chess is one example. On the first move each player and has 20 options, thus after a single move by

each player the positions on the board have increased to 400. One solution to this problem is to try and develop a systematic approach to searching by looking at the chess pieces if there are good or bad moves. The lesson is that, given a developing search space that has a number of nodes available for expansion, we should expand that node which seems the most promising or in other words the node that seems closer to the goal. This idea underlies the premise of the heuristic search. Heuristics being the means by which we estimate which will be the best node to expand. There is another level of complexity associated with finding the best plans which has to do with finding the shortest possible solution. The method presented in chapter 4 is concerned with exploring all the states of the search space. Therefore although we acknowledge the need for employing searching mechanisms that can calculate distances from a given goal state, in this approach this is not considered to be necessary.

As well as ordering the nodes we can order the operators we will apply to a given state. For example a chess player may have a rule the knight should always be moved when possible and begin the search by looking for possible knight moves. Therefore, a heuristic search can be ordered on state evaluation, where nodes are evaluating it, or on operator evaluation or on a combination of the two, which is most common.

An excellent example of a heuristic search is the 8-puzzle (taken from both children's toy the 15-puzzle) which consists of eight sliding tiles in a 3 x 3 grid. The goal is, given a random starting position, to rearrange the tiles by sliding them into the vacant position to attain the correct configuration. Because it is too complicated to be solved by brute force methods alone it is perfect to exemplify heuristic methods.

The key problem of the heuristic method is to develop an evaluation function that is able to assess promising nodes in the developing search space. There are numerous evaluation functions we could apply to the 8x8 puzzle. The first type of evaluation, which is very simple, would be to count the number of misplaced tiles. However this may prove to be unfruitful since the score can be improved simply by moving one tile to the correct location and possibly blocking the path of another tile. A second way of evaluation would

be to count and sum the distances of each tile to the proper locations. This however would also fail due to problems mentioned above, but it does have the advantage of recognising that not all of the good moves result in a tile being placed correctly. A good refinement would be to add to the sum resulting from the last suggestion the number of tiles not followed by their successors.

All of the above evaluative functions would allow a node to be assigned a number or score. However what happens if two nodes have the same score but one takes a longer amount of time to reach. It is important therefore to expand the node that can be reached by the shorter path, and score nodes by adding the cost of reaching them to the score of the evaluation. If we let the evaluation function return an estimate of the number of steps required to move from the node being evaluated to the goal node, then this combined with the number of steps taken to reach the current node, will represent the length of the path from the initial state to the goal state. The lower the score the more efficient the search. This was a very popular idea and there was a lot of work, especially in mathematics carried out exploring the resulting search algorithm and especially relating to whether the algorithm would result in finding a solution using the shortest path to a solution. It is possible to prove that if the evaluation function represents a lower bound on the actual number of steps to the goal, then the solution we find using the algorithm would be optimal.

It must be stressed that finding an evaluation function is difficult in reality, and may not lead to an optimal search if we are more interested in minimising the number of nodes examined in the finding of the solution than in the number of nodes on the eventual path found. This can be seen by considering that an evaluation function that returns zero in all cases will represent a lower bound, and will find the optimal solution, yet it will generate a breadth-first search. The aim is to find the most efficient search. Using the 8 x 8 puzzle, as an example, a good evaluative function proved to be the sum of the distances from the home square plus 6 times the number of tiles not followed by their successors, plus 3 if there was a tile in the centre. This did not represent a lower bound on the steps from node to goal, and so the solutions it found were not provably optimal. The moral of this is that

while we do want to find that solution, in reality, we will accept for finding a solution whilst expanding a practicable number of nodes. The best strategy to adopt was invariably bound up with the nature of the problem – the size of the search space, the branching factors, the likely depth needed to find a solution, and the availability of an effective evaluation function. This particular example may have moved slightly away from the scope of the thesis but it is considered essential as it provides a good understanding of combining heuristic search algorithms. In fact all the case studies examined in chapter 4 had a limited search space which would make it possible for either technique (breadth first-depth first) to be equally successful.

5.2.2 Limitations of search

It is worth mentioning here –although slightly out of the thesis scope- that as the subject of AI developed, search alone was recognised not to be enough. There were several factors that made it unsatisfactory as a general problem-solving technique. Search size space can be limitless and thus cause many problems. If we think of the 8 x 8 puzzle we looked at earlier imagine the same problem on a larger scale such as the 15 x 15 puzzle or even the Rubik's cube. For most problems the search space is too large to enable an efficient search program to solve them in an acceptable time. Problem-solving programs such as chess still rely heavily on search, searching not for a solution but an acceptable move, so we look as far forward as time allows and return the best move we can find.

Other problems with heuristic searches, (apart from the problem of the size of the search space) were mainly to do with finding an appropriate evaluative function. Evaluative functions were not simply difficult to find but they were difficult to establish by any other means than trial and error and usually turned out to be specific to the problem. Thus the lessons learned from the 8 puzzle were of little interest unless they could be used on the 15 puzzle. The whole approach fell into difficulties since the evaluation function failed to lead to a solution, but emphasis was placed on local optima. Also there is no smooth progression from initial state to goal-state and in such cases there would be no effective way of using search method to find a solution.

This approach declined in popularity because in AI research, we're in a sense interested for a computational model of the mind, and such an approach does not attempt to model human reasoning, when confronted with such problems. Humans are unlikely to systematically search, except in the simplest cases, to find a solution. Certainly no human would use the solution suggested to 8 x 8 puzzle. This approach, in some motivations, runs contrast by essentially using non-intelligent methods to solve problems, and thus reduced problems from requiring intelligence to those that can be solved mechanically. It was for these reasons that an interest arose in human problem solving.

There are three basic types of human problem solving techniques, however much of the work in AI would not draw these distinctions but would label them all "heuristic" to cover all three domains. These three domains are decomposition of problems, search for patterns and rules of thumb.

5.3 Heuristic methods

In the following paragraphs we examine three different heuristic methods. All three steps to problem solving examined -in the following paragraphs- were performed during the development of our approach as it is explained in chapter 4. The problem was decomposed into states, premises and restrictions that described the initial picture. The conversation theory that was applied for the purpose of animating the model is based on patterns developed by examining several case studies. The PAD rules that defined the boundaries of the movement could also be thought of a series of rules of thumb. AI literature has acknowledged these three steps as the basic types of heuristics regarding human problem solving methods. Since we have already related these steps to our approach, the following paragraphs will briefly outline the three steps (decomposition, patterns and rules) from an AI perspective.

5.3.1 Decomposition of problems

One of the tendencies of humans when confronted by complex problems is to decompose them into smaller, more manageable, problems. These smaller problems may be simple enough to tackle straight away or they may need to be further decomposed.

A typical way of attempting to solve a Rubik's cube at first go is to get one face correct, then a second face, then a third and so on. This shows that there are numerous ways to decompose a problem, some of which will be more effective than others. Continuing with the Rubik's cube for an example, the step of getting one face correct and then moving onto others is not very effective. A better way is to get the corners correct first and then attempt the others pieces. One reason why the Rubik's cube is such a hard puzzle is because seemingly natural decomposition can be counter productive.

Another example could be using the 8 x 8 puzzle where an initial decomposition may be position 1, 2 ,3 and so on, although it would be more effective to get the first row right, then the first column, and then shuffle the final 3 pieces around until they are correct. This knowledge, once discovered, can be transferred onto other puzzles such as the 15 x 15 puzzle by completing the first row, the second row and then the first column and then shuffling around the final three pieces until it is correct. This knowledge can also be passed onto other types of puzzle such as the N by M grid. You get the first N-2 rows right, then get the first M-2 columns right and then shuffle the last three pieces. Yet, it should be noted that getting the first row right also involves some decomposition, by getting 2 to follow 1 and 3 to follow 2 and so on. This technique can also be generalised.

It is instructive to compare the way the problem can be decomposed with the final evaluation function given above. Obviously, following the strategy outlined here will tend to improve the result from the evaluation function both by getting tiles to their home squares and by increasing the number of tiles followed by their successors. The human problem solver is not concerned to find the optimal solution, but with knowing at all times what he is doing, and can see a steady flow of achievements as he completes subtasks which he knows are steps on the road to a complete solution. It is this kind of factor, involving a strategic rather than a purely tactical appreciation, which avoids the kind of problems encountered by AI problem solving methods. Knowing how to decompose the task into the appropriate subtasks gives us a strategy for solving the problem, but it is unlikely to take us all the way to a solution. We still need to know how

to operate at the tactical level, so that we can achieve the subtasks and this is where the second feature of human problem comes in. It is the same feature that was addressed during this thesis in the form of the conversation theory.

5.3.2 Search for patterns

As well as decomposition, another human trait involved in problem solving is identifying patterns which they are familiar with and know how to respond to. The skilled problem solver will have knowledge of many such patterns and appropriate move sequences, and he would solve the problem by decomposing it into subtasks where such patterns can be identified and appropriate sequences performed. This is also the way to go about solving the Rubik's cube, where the whole trick of mastering the cube is to know a series of moves which will exchange two particular pieces, or re-orientate a piece, whilst leaving the other parts of the cube unchanged. The two-part solution method is quite general: decomposition supplies the strategic knowledge of how to go about solving the problem, reducing it to a set of tasks which can be performed by recognising patterns to which the appropriate actions can be taken.

5.3.3 Rules of Thumb

This third human problem-solving characteristic, which can be seen, is that of "rules of thumb", which refers to actions which are usually, (not always), good things to do in certain circumstances. They are quite similar to patterns referred to already, however rules of thumb should be applied more opportunistically rather than an execution as part of a design, and most importantly recognised as being fallible. A rule of thumb relating to the 8 puzzle, is that if a tile is in the central position, it would be a good idea to move it, because it increases the number of moves available, and it also improves the evaluation function used above. Within the current framework, a rule of thumb should only be used as an effort to reach a familiar pattern where none can be seen. A general notion to be introduced is that, a heuristic is used to cover all three kinds of problem-solving knowledge. A heuristic can be seen as a matter of recognising a situation and making a certain response to that situation. The situation may be simple and the response uncertain in effect, as in the case of "rules of thumb" above; or the situation -and the response to it-

may be more complicated and the effect of the response more certain, as in the case of what termed "patterns". Again the situation may be a task to perform, and the response to perform a series of subtasks, as in "problem decomposition". This way of seeing things suggests that we can represent all three types of problem-solving knowledge in the same way. In particular we can represent all three types of knowledge in the following way using a uniform mode of expression:

Rules of thumb such as "If tile in the centre, then generally move that tile" or a PAD rule such as 'Relationships between agents of initial value s[on] m[off] can be turned transformed to s[on] m[on] or s[off] m[off] or s[off] m[off]. Control is passed to the agent with higher authority while capability is positive only when mutuality is on'.

Patterns such as "If pattern A obtains and goal is goal 2, then perform move sequence 6" or an enterprise thesaurus rule such as 'Agents that belong to the same structure are assigned equal control and positive capability'

Note the use of the qualifier on the action in the case of the rule of thumb whereas the pattern is thought to be always applicable. Finally we can represent decomposition as

"If goal 1 then achieve subgoal A and subgoal B"

An attempt to follow through this method of knowledge representation provides us with the first major knowledge representation paradigm that we shall examine in the following section. In the next section we discuss knowledge representation as well as the available technology for realising such complex data structures.

5.4 Knowledge engineering

Another question that chapter 4 may have raised is 'how are rules constructed' and 'how are they stored so that we can make successful inferences'. Indeed the relationships between these types of rules are far to complex to be represented by a relational database

structure. On the other hand these rules do not just represent relations between data but knowledge and facts about models.

5.4.1 Background

Knowledge representation is an integral and important part of knowledge based systems. Knowledge must be explicitly processed via such mediums as programming, languages and diagrams, in order for this knowledge to be processed and hence solve problems [Newell 1982]. There is a division between knowledge modelling and selection of appropriate knowledge representation language to represent reasoning as well as the knowledge itself [Davis 1993].

Even with the same knowledge source, engineers, due to their specific interests and aims in modelling domain knowledge, frequently produce different models of KBS. All representations are approximations of reality, and are thus flawed. According to [Clancey 1993] knowledge representation is an activity of modelling knowledge in order to create a new knowledge base. Any approximation will include some things and exclude others, and through selecting what to include we are choosing how to see the world and what to see in the world. This also applies to the approach presented in chapter 4.

Facts or beliefs can be considered as data structures that contain the problem and process it into a form in order to find the solution. Data structures can be divided into two sections; representation of the problem and representation of the problem solving knowledge [Newell 1982]. The distinction between knowledge and representation is important both for designing and analysing knowledge based systems. During the design process it is possible to enable what knowledge is required to obtain a level of competence without commitment to one representation. Once the type of knowledge required has been decided by designers, specific representations can be chosen to encode this knowledge.

If the emphasis is on knowledge acquisition system, competence is a priority, whereas when choosing a representation performance, as it is many times the case in enterprise

modelling, then performance measures are of uppermost importance. Representation language comprises of syntax (meaning) and symbols. The semantic interpretation is accomplished through algorithms that access and infer information. From a syntax point of view the most important aspect is how the information is stored in an explicit format. In contrast from the inferential point of view the most important aspect is the way in which explicit information can be used to derive information implicit in it.

[Davis 1993] describes the combination of language and interpretation, as a double-sided coin that can not be usually separated. "A knowledge representation language, however like logic, is not equal to knowledge itself" [Newell 1982].

"Knowledge representation is most fundamentally a surrogate, a substitute for the thing itself, that is used to enable an entity to determine consequences by thinking rather than acting, that is, by reasoning" [Davis 1993].

There are a variety of arguments used to support syntax over inferential aspects and vice versa. Arguments supporting the syntax aspect stress the naturalness and expressiveness supported by the knowledge representation language. Other arguments supporting inferential aspects will draw attention to the power of underlying inference mechanism. Using different knowledge representation languages results in different types of systems. In fact even the same type of systems, due to different purposes, can be represented in different languages to achieve more efficient performance or expression.

As well as the schism between syntax and inference aspects, knowledge representation language can be divided into five levels, and these levels could help determine which language is appropriate for the representation of a specific domain knowledge. These levels correspond to the series of semantic networks [Brachman 1985] used for knowledge representation. This type of structure was first introduced as a Ph.D. thesis [Quillian 1966] as an attempt to model human memory via a relational network of semantic memory. In Quilllian's thesis words were represented as a network of nodes which represented word concepts and links between nodes which represented definitions.

The same structure was again later used [Fillmore 1968] in the processing of natural language. Both methods attempted a knowledge representation within the domain of linguistics. Other attempts based on this model were also made in the area of language understanding [Heidorn 1972] as well as modelling organisational structure, computing networks, distributed systems etc. In this thesis we consider the 5-level semantic net as a knowledge representation mechanism that allows us to distinguish between expressions, data structures, symbols and definitions.

5.4.2 Implementation level

The most important aspect at this level is designing data structures to accommodate knowledge, and to discover and implement appropriate algorithms that allow for the desired effect. Secondly another aspect which is of high priority to this level is to increase the speed at which the data is processed using an appropriate indexing mechanism. Although every representation must be implemented in the system by a data structure, the actual representational property lies in its correspondence to something in the real world and the limitations imposed by the correspondence.

5.4.3 Logical level

At this level the knowledge representation language should provide sufficient logical properties to represent knowledge. Two major syntactic concerns which are meanings of expressions in formalism (Limited interpretation of expression), and expressive power of formalism (abilities of expressions). From the inferential point of view, the main concern at the logical level is the logical properties of the inference procedures.

5.4.4 Epistemological level

This level deals with types of knowledge structuring primitives that are needed. Emphasis at this level is the set of generic primitives that allow us to describe the objects of the world as formal structural units and their interrelationships. These are called knowledge structuring primitives, which refers to types of primitives required to represent the knowledge and types of inference strategy to be made available.

5.4.5 Ontological Level

The Ontological level is concerned with meaning [Guarino 1993]. A knowledge representations is a set of ontological commitments, because the primitives represent an agreed meaning between users [Davis 1993]. The focus at the ontological level is the explicit specification of the meaning of primitives in a language.

5.4.6 Conceptual level

This level is concerned with primitives to be included in a knowledge representational language. This is a broad overview of the types of primitives that go best with certain levels in order to be able to choose which knowledge representation language is appropriate for a specific type of knowledge.

Having looked at the five distinct levels of knowledge representation the chapter concludes with an account of the different types of language available for producing knowledge bases.

5.5 Frame based expert shell system

A number of expert system shells attempt to amalgamate various representational formalisms in a new mixed knowledge representation language in order to explore their advantages. These systems provide a large number of different knowledge representation languages, object-oriented programming languages and various inference machinery to be integrated in a coherent environment. The three most popular types of languages are procedural languages (object oriented languages), frame based languages and production rules embedded in shells.

5.5.1 Frame based formalisms

These languages inspired the representation method for our approach. They are built by utilising different programming languages such as C, C++, Lisp and so on, which allow the user to initiate different types of knowledge using the procedural languages which are housed in a frame called 'procedural attachment'. A frame is a data structure to represent

a 'stereotyped' or expected situation and they are the primary importance in frame-based languages.

A frame contains a system of slots to characterise its properties. There are different types of frame, such as the super-frame and its descendant the sub-frame, and thus can inherit properties from super-frames. The relationship between the two is called a 'has-a' relationship; an example of a frame. The instance is a terminal of the frame data structure, so it is unable to have a 'has-a' relationship with other instances or frames. Descriptions within frames are called slots, and they consist of two parts; a slot name (an attribute) and a slot filler (value for the attribute or any restriction on range of possible values. This structure was used to develop the enterprise thesaurus presented in chapter 4.

5.5.2 Rule-Based formalisms

We discussed rule-based formalisms earlier in the thesis as a possible way of representing facts and guiding a model's movements. IF-THEN or production rules comprise of the antecedent the IF-part of the rule and a consequent the THEN part of the rule. The IF part of the rule must be satisfied in order to enable the THEN part of the rules. An example of which is:

If cond1 and cond2 and...
THEN action1 and action2...

If the premise associated with a rule matches the current state of the working memory, then the actions are ready to be executed. The working memory is used for storing 'temporary' information relating only to the task at hand, and is an important component in a production rule. In a mixed knowledge representation expert system shell, rules can be triggered by, and manipulate the values in the frame. The following example portrays the syntax of a rule in KAPPA-PC.

GoodElecsys[car/Autos]

If

car: Spark #= Good And

car: Timing #= InSynch And

car: Battery #=Charged

Then

car: ElcSystem =Good;

GoodElecsys is a rule name and the car variable represents a class called Autos. In this case, the Autos frame includes Spark, Timing, Battery and ElcSystem slots. The Spark, Timing and the Battery are preconditions of the rule and the ElcSystem is the action part that includes the assignment of a value to the ElcSystem slot. Some shells, such as KAPPA and Goldworks, provide rule sets to group related rules into a set. In KAPPA, the rule sets can be specified and stored in a global slot. Sets of rules can be interpreted in a backward and forward chaining fashion. Rules can also be assigned different numbers of priority. Setting the priority of a rule allows the order of precedence in the reasoning path when more than one rule applies. The syntax of rules and frames in different frame-based expert system shells has it's own format. Even though some of them might be similar, they are not compatible. Thus it requires a translation mechanism to translate one format to the other [Hruang, 1996].

5.6 Application Programming Interface in Expert System Shell

The following paragraphs provide a bit more detail into the available technology for building knowledge based systems for enterprise modelling. All system expert shells are implemented by programming languages as modules. The functions provided by the shells are controlled by the specific APIs. The APIs are associated with a runtime library that is provided by the expert systems shell to enable the applications built by the shells to integrate with other applications. For example, KAPPA-PC is implemented by C. The built-in functions in the KAPPA-PC are modulised onto a set of runtime libraries such as Dynamic Link Library (DLL) to allow the applications to embed or be embedded with other applications held in different environments. Thus other MS-Windows applications

can interact with KAPPA-PC applications via KAPPA-PC APIs, and C or C++ programming languages.

Like KAPPA-PC, ILOG RULE is another object base expert system shell, but it is implemented in the fashion as an object-oriented method using C++. The applications designed with ILOG RULE can also be controlled with specific APIs. The rule bases in the ILOG RULE are implemented as a set of C++ class, so the generated C++ class have their APIs strongly related to C++ classes of the ILOG RULE runtime library. As a result, other applications can communicated with the applications built-in the ILOG RULE.

AionDS (ADSAPI) provides an open architecture that enables the user to integrate knowledge bases with other applications in various ways. The ADSAPI allows the user to call the knowledge base built in the AionDS from other applications, pass the input, and receive result back. In other words, the inference engine AionDS can be embedded into a large application. The user has to include two files in order to activate the ADSAPI. These are a library file known as ADSAPIW.DLL containing the ADSAPI dynamic link library routines and the ADSAPIW.LIB file that is the import library used to link the applications that use ADSAPI.C or C++.

There are also a number of expert system shells for example, OBJECT IQ, NEXPERT, Crystal etc. that provide the APIs for the user to integrate their knowledge bases with other applications. Although they are not illustrated here, they enable the user to retrieve the knowledge in the built-in shell and construct a new knowledge base by using these facilities without getting into details of software internal mechanism.

5.7 Conclusions

This chapter tried to answer some of the issues that chapter 4 raised. The method presented in chapter 4 was applied in three different case studies including the health service, a brokerage system and part of the automotive sector. Models were presented

initially in a diagrammatic form and later decoded using lisp type lists. The lists were processed to reveal a new state of the animated model.

The required state was expressed in a list called action. In the case studies the desired state was only a step away from reaching the goal expressed in an action list. However when multiple movements were required we had to employ heuristic methods in order to identify the correct state and more importantly the path that led to that state.

Searching methods can either be depth first of breadth first. Both explore the search space defined by the model and expand each state by creating or deleting roles and relationships. The differences of the two lie in the way search in made. While depth first search explores each agent to all the level that it can be expanded, breadth first expands all agents one level simultaneously. However AI literature has taught us that these searches may need to be combined with further paradigms in order to be more accurate. We examined problem decomposition as opposed to search for patterns and rules of thumb. Both methods were used to define our approach in chapter 4. From their descriptions one could notice the similarities between the approach described here and the approaches for developing business or workflow models. Although there are similarities between this research and business or workflow modelling tools, there are some fundamental differences that are worth pointing out. Business modelling tools in general are capable of developing ontological descriptions of enterprise systems. BSDM (business structured domain modelling) [King 1995] by IBM or ART by inference Corporation [eGain 2001] develop ontologies using two conceptual components namely the *entity* and the *link*. Organisational structures can be represented by a series of links between entities. Some of these tools employ case-based reasoning in order to apply past solutions to existing problems. If for example we take an online bookstore and try to develop a business model using BSDM or ART we would expect certain entities such as *customer*, *order* or *publisher* to be included in the model. Some relationship patterns between these entities may also exist. Business modelling tools are capable of recognising these similarities between past and current scenarios and are able of applying past solutions of current problems.

Workflow systems such as WADE provide operational semantics to process descriptions. In a word, they execute process models. Different workflow engines execute different process models. Different process models focus on different aspects of processes. Some are designed for complex decision-making and therefore encapsulate rules and logic needed for decision support, while others work with relatively routine and recurring processes and incorporate greater detail about individual activities within the process. Workflow systems may also provide varying levels of both exception handling, monitoring and management capabilities. There is no workflow system that is equally strong in all areas [Bussler 1994].

Both types of tools are capable of developing organisational descriptions, process description and address information flows. This thesis however is concerned with the nature of the links between entities and how these links can evolve. More importantly the approach in this thesis as opposed to business modelling and workflow tools is referring to a different stage of the life cycle. Business modelling techniques for example usually refer to existing systems or new systems after their requirements have been finalised. The approach described in this thesis is concerned with the exploration of the problem space when the constraints and the controls that affect the relationships between agents, or the links between entities are not yet known. Both types of business or workflow modelling require some of the requirements of the system to be known. The approach described in this thesis is applied at the very early stages of the life cycle when requirements are not yet specified.

The same is true for model checkers. As explained earlier it is really necessary to maintain track of movement. One may want to establish all possible relationships a particular agent can be involved. In this case particular heuristic mechanisms have to be employed to expand the agent all the possible states. Alternatively one may wish to expand all agents simultaneously. Model checkers maintain their heuristic mechanisms hidden from the user. In our case this is essential. Additionally model checkers verify a

formula that can be thought of as a goal state. In our case we are not looking for goal states. We are interested in the evolution of relationships before we reach a certain state.

The chapter also discusses the complexity of developing knowledge-based structures, and the requirements of representing knowledge. The frame-based structures discussed in chapter 4 are explained in more detail in terms of rules and formalisms. The methods, structures and data representations presented in this chapter can be realised using environments such as KAPPA-PC, ILOG RULE and Crystal.

Chapter 6 Implementation

6.0 Introduction

Some of ideas developed during this research were tested by the development of a prototype enterprise modeling system. In chapter 3 we divided enterprise models in structure, process and resource oriented types. Each of these models draws attention to different aspects of the enterprise. The following chapter provides a detailed description of a resource oriented model that was developed as part of this thesis. It allows the modeller to build a diagrammatic representation of a supply chain and identify the links between the various agents. Furthermore it provides the modeller with information regarding the link and finally provides a mechanism for assessing the impact of deleting or creating new links. The impact is calculated in terms of resources been supplied by or to the various agents.

The prototype is based on a relational database and a series of algorithms which can display the data in a graphical form, assess the impact of new data added or deleted and displays the results in a text based format. In this chapter I describe the implementation of the ideas (mainly the thesaurus) outlined throughout the thesis. The development process was divided into the following stages.

- Formation of data structures using relational database structure (Enterprise Thesaurus)
- Development of Interface
- Development of Algorithms to assess impact of change
- Development of Queries

6.1 The prototype

The initial step was to design the database using Access 2.0. There are two basic tables:

- one describing the nature of each company, regardless of the level on which they belong and,
- one providing details of the relationships between the companies themselves.

This structure simulates the thesaurus structure proposed in the previous chapter. *Normalisation* was used during the design process to structure the database in a flexible

way so that further developments on the same prototype could take place. The two basic tables, *Company* and *Relationships* are complemented by the tables *Status*, *Link_Type*, *Offer*, *Queue*, *Company_Type*, *Query* and *Decision*. The contents of the tables can dynamically change in order to reflect the state of the model at all times.

6.1.1 The Database

The first table holds information regarding the end product produced, the annual capacity as well as the control capacity, the number of employees and finally the location of the main production plant. The field *status* indicates the nature of the company and the level (the level in the enterprise network) to which the company belongs. In the *Company* record the status is represented by a number. The number represents the primary key used to access the *Status* table.

Name	*Text*
Status	Integer
Product	Text
Product Capacity	Long Integer
Control Capacity	Long Integer
Service	Text
Employees	Integer
Base	Text

Table 6.1 Company Table

The *Status* table describes the nature of the company. It is almost identical to the *Link_Type* table (described later) although it serves a different purpose. The first field is a primary key whereas the second describes the company, which can either be an integrator or a component manufacturer.

The *Relationships* table holds information regarding the links between the companies stored in the database. The table is used for cross referencing data. It can be thought of as the table that holds information about predicate values 'uses', 'supplies-to' and supplied-by'. The links are described only by numbering system. The fields *From* and *To* hold numbers which correspond to company names and are primary keys for the *Company* table. The field *Start* indicates which company has initiated the creation of the link, whereas the field *End* declares who the other party is. The field *Type* declares the nature of the relationships. This field declares the buyer and the supplier. The field is also numeric and is used as a primary key to open the fourth table of the database.

Company From	Integer
Company To	Integer
Type	Integer
Date	Date

Table 6.2 Relationships Table

The *Link_Type* has two fields. The first one is a number, the primary key, and the second is a description. The field *description* explains the nature of the relationship. In the prototype this has two options: *buys from* or *sells to*. More options will be added to the database as the prototype expands and more companies with different attributes are added.

6.1.2 The Interface

The Interface was developed using Visual Basic Professional. It is a Windows based interface that makes use of many Window's functions as well as icons, menus and pointers. All the functions of the interface conform to the common Microsoft windows interface procedures. The systems database can be manipulated by using *Microsoft Access* or *Visual Basic 3.0 Data Manager*.

6.1.3 Features

The prototype offers the following options:

- Database handling functions such as Add, Delete and Update records.
- Graphical presentation of the relationships between integrators and component manufacturers.
- Production of reports regarding the details of the relationships.
- Assessment of broken relationships and decision support regarding alternative options.
- Presentation of enterprise relationships displayed on several maps of the U.K.

Figure 6.1 shows the screen that enables database handling. The buttons show the functions supported

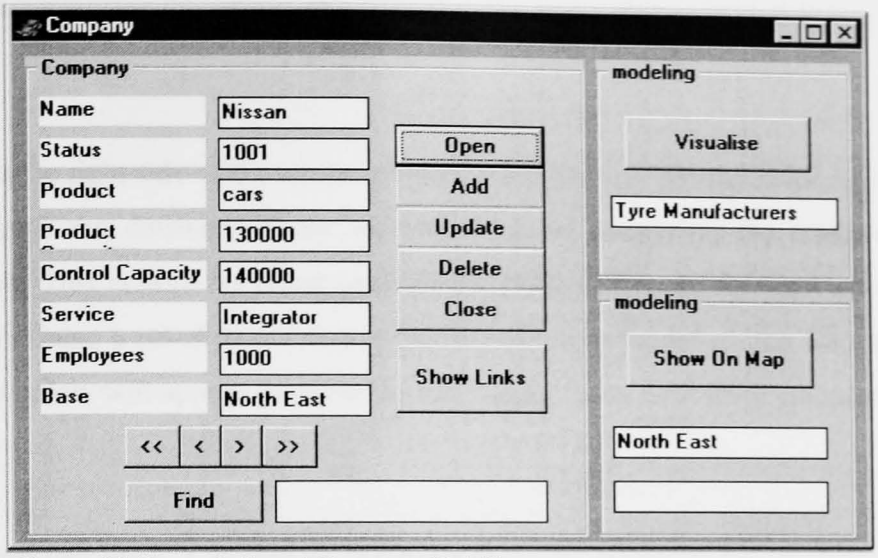

Figure 6.1 Presentation of Company Details

Figure 6.1 shows how data are retrieved from the thesaurus and presented on the user's screen. What we have here are the details of a car manufacturer based in the northeast England. The first field *Name,* is self explanatory. The field *Status* shows a number which indicates the level the company belongs to (*integrator, component manufacturer, etc.)*

Nissan in the example has been assigned the number *1001.* The numbers are associated with each level of the supply chain. (1001 represents top level/integrator. 2001 represents component manufacturer). Numbers have replaced the text based description *(Status: Integrator)* or (*Status: Component Manufacturer)* to assist the normalization process and enhance file handling procedures. The field *Product* shows the product's name in a text based form. The fields *Product Capacity and Control Capacity* are there to indicate the number of items produced every year as well as the maximum number of items that can be produced, in case of emergency. The *Service* field is there to show a description of the service the enterprise provides in text based form. The fields *Employs* and *Base* show the number of employees and the location of the enterprise respectively.

Buttons are provided to perform the basic file handling operations such as *open database, add new record, delete record, update file and close database.* Additional buttons have been added in order to make the navigation process easier and straight forward. The *Find* function allows the user to search for a particular enterprise by entering its name in the text box. Should the company exist in the database the details would appear on the screen,

otherwise a message would inform the user that the name does not match any of the records.

The options at the right side of the screen are used when the user wishes to see a graphical representation of the database. There are two ways of presenting the database in a graphical form. This is done by either pressing the button *Show* on the right side of the screen or by defining a set of parameters at the text boxes on the left. If the first option is taken a new window will appear showing the enterprises' links with other enterprises.(see Figure 6.2)

In order to present such a graphical representation the database has to find all the company records that are related to the enterprise being modeled. The second option assists to narrow the search and present the results in a more organized way. The parameter under the heading *visualize* indicates the category of companies that the user wants to see on the screen. Companies are categorized according to their product. In the example given in Figure 6.1, the user typed *tyre manufacturers*. That means that only tyre manufacturers will be linked with the box in the middle of the screen that carries the name of the enterprise being modeled. The user could narrow the search even further by declaring the area of the region which he wants to model. In the example the selected region is northeast. This means that the diagram will show the links the enterprise has with tyre manufacturers in the northeast only. All sorts of combinations can be made using these parameters.

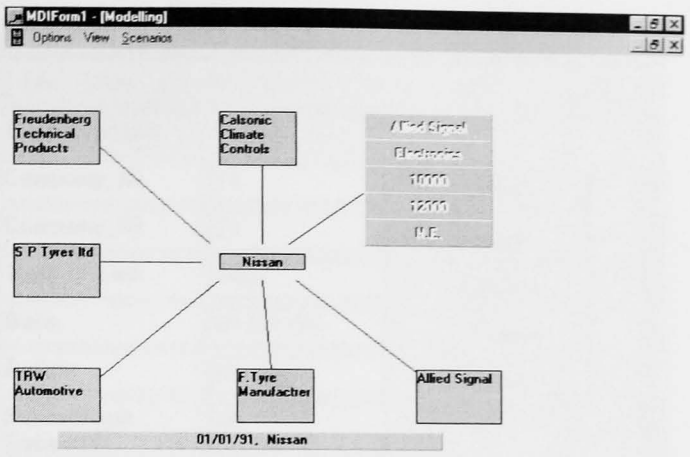

Figure 6.2 Model of Nissan (Prototype system)

Figure 6.2 shows a graphical representation of Nissan using the Enterprise modeling prototype. In the middle of the screen we can see Nissan surrounded by component integrators. All the companies that belong to the same level, are represented by the same box. In this case all the companies are component manufacturers and are represented by rectangular boxes. Information appears in the top left corner showing the details of these companies. This happens whenever the user moves the cursor onto one of the icons. Recall that the boxes represent companies Nissan is related to.

Every time the mouse pointer is moved over a box, like the one shown in the Figure 6.2, a box appears and information regarding the highlighted company is displayed. Figure 6.2 above shows that the highlighted company is *Allied Signal* that produces electronics at a certain production capacity and control capacity rate. It also confirms that the company is located in the northeast. The bar at the bottom of the screen indicates when the link was actually established and who was the author. There are a number of options related to the screen presented in Figure 6.2. The impact of both links being either deleted or added will be shown on the screen in text format. Details of how this is achieved is shown in the example in section 6.2. None of the changes (delete or add company) will be registered to the database unless the third option *'update database'* is selected. At this point all changes will be recorded automatically in the thesaurus. Figure 6.3 shows how the table that registers links looks like.

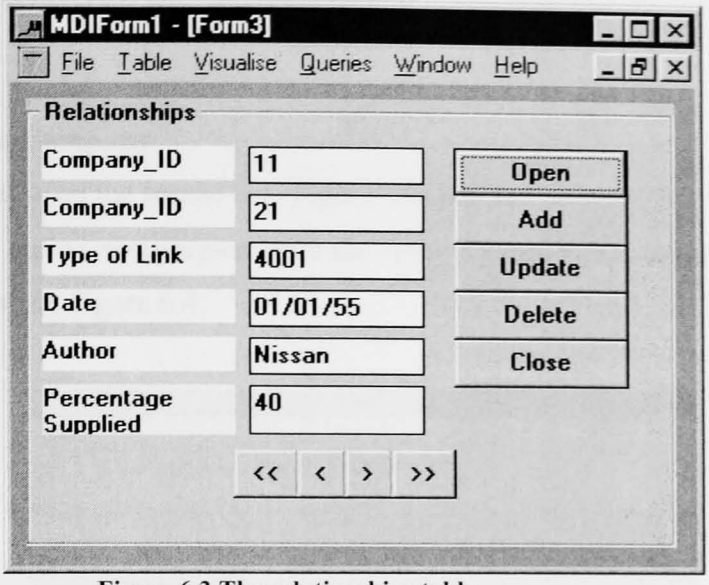

Figure 6.3 The relationships table

Relationships can either be added manually using Microsoft Access or the Visual Basic Data Manager. The information in the table provided is used in conjunction with the other tables of the prototype. The first two fields (company_id) show the code number of the enterprises that are linked (11 and 21). The first one shows the company that initiated the link. This is used to assist the algorithms to identify what type of link has been established, as is declared in the third field. The most common types of links are *'Buys From'* and *'Sells to'*. These correspond with the 'supplies-to' and 'supplied-by' predicates. In the example given in the previous paragraph, if Nissan was the company that was being modeled, and Allied Signal was put on the diagram manually, then the system would assume that Nissan initiated the link and would assign a *'Buys from'* type of link to the record. The two remaining fields are self explanatory. The first declares the date the link was created and the second the author. The section that follows gives an example of the modeling process and how the implications of a company being added or deleted are assessed.

6.2 Case Study

The example that follows aims to demonstrate the capabilities of the prototype. In order to maintain consistency throughout the example the author has decided to model *Nissan*. The example continues from Figure 6.2. Assume that *Nissan* is the company selected by the database and that the user has pressed the *Show Links* button at the initial screen. This

would result in a diagram with a *Nissan* box in the middle and the suppliers surrounding it. (see Figure 6.1)

6.2.1 Deleting a Company

When the model is on the screen (i.e. Figure 6.2) the user is presented with a different set of options. The option *show report* from the options menu will result in the production of a report as shown in Figure 6.4.

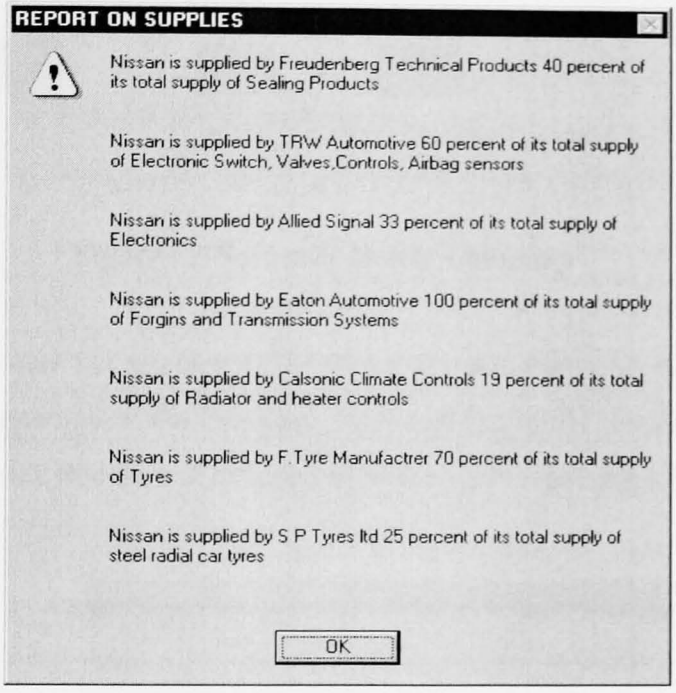

Figure 6.4 Report on the model

The algorithm locates the details of each of the companies displayed on the screen, calculates the capacity they supply to Nissan and displays a full report. If the option *Delete Company* is selected from the *Scenarios* option then the system will require the user to enter the name of the company he/she wishes to delete from the model. Figure 6.5 illustrates the process.

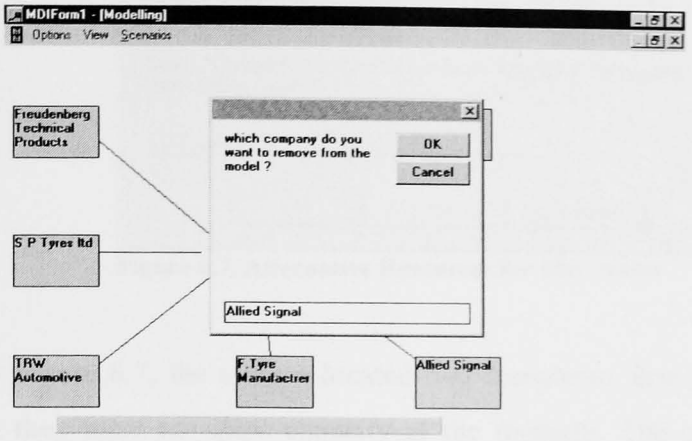

Figure 6.5 Scenario–Delete Company

In Figure 6.5 the user has selected *Allied Signal*. By pressing *enter* the system will assess the impact the departure of the company. Although the initial model will remain on the screen the box in the middle will be replaced with a report showing a list of the companies that will be directly affected, as shown in Figure 6.5

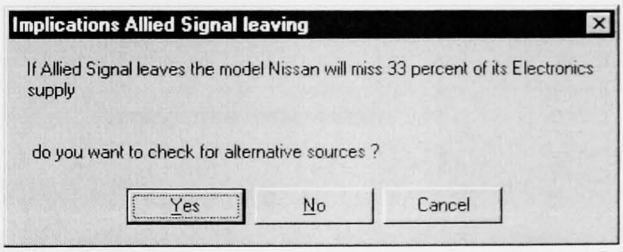

Figure 6.6 Implication of Allied Signal leaving the model

At this point, Figure 6.6 suggests, the user is presented with three options (yes/no/cancel). If the user presses the *YES* button the system will try to locate other companies that supply electronics and report back as Figure 6.7 shows.

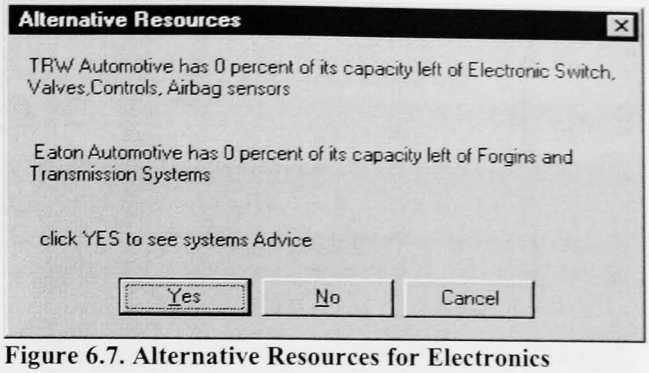

Figure 6.7. Alternative Resources for Electronics

According to Figure 6.7, the system located two companies that supply electronics but unfortunately they have no spare capacity at the moment. The user is presented with another question at this point. Selecting *YES* will result in the reply that follows whereas any other reply will take the user back to Figure 6.5. All *CANCEL* selection at this stage will lead back to Figure 6.2.

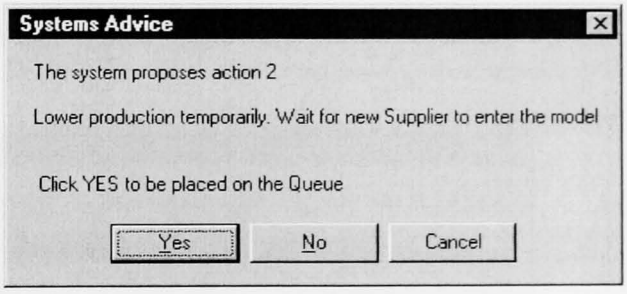

Figure 6.8 Systems Advice

Figure 6.8 show the systems advice accompanied with a question. The system is capable of storing information of enterprises that are in need of supplies. The details of the enterprise are placed in "Queue" file as shown in Figure 6.8. The system has proposed action 2. There are in total 9 actions that were derived from the conversation theory tables 1 and 2 described in chapter 4. If the enterprise is in the queue, whenever a new enterprises with spare enters the model the system will produce a list with the companies that are in need of the new supplies.

6.2.2 Adding a Company

The process of adding a company to the model is identical to the process of deleting an enterprise. From that point onwards the user is presented with a different set of options. Assume, for the sake of consistency, that there is a model of *Nissan* on the screen again as

shown in figure, and the user selects the option *Add Company* from the *Scenarios* menu. Figure 6.9 shows the screen that follows after that selection.

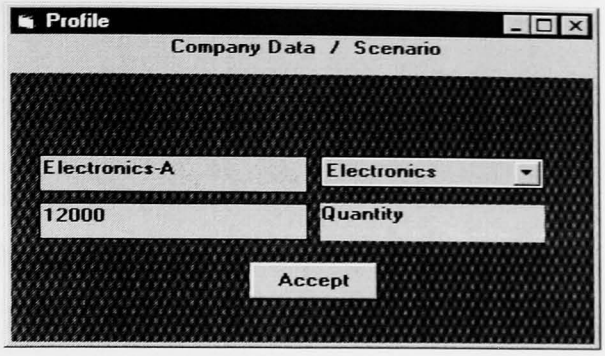

Figure 6.9 Adding a Company

The screen requests the user to enter information regarding the profile of the company to be added. The information relates to the name and capacity of the company as shown in Figure 6.9. When the information has been entered the system will respond with a list of the companies in need of electronics. Figure 6.10 shows this list.

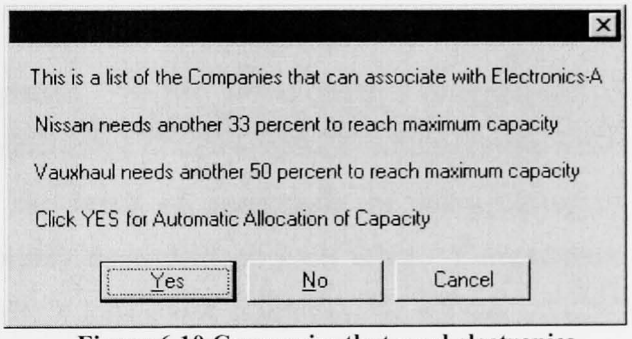

Figure 6.10 Companies that need electronics

The list shows which companies in the database have been placed in the queue, waiting for a new supplier of electronics. The system is also proposing automatic allocation of the electronics capacity this will be based on the priority given to each company in the waiting list. Priorities can be given by the user, or determined by the system by estimating the percentage of supply needed. If *YES* is selected the system will automatically allocate the capacity as show in figure 6.11.

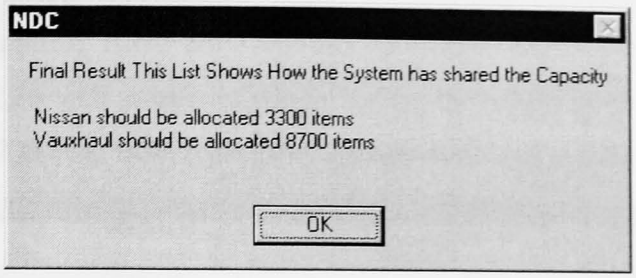

Figure 6.11 Automatic Allocation of New Supply

The features illustrated by the example are accompanied by a set of algorithms which ensure that consistency is maintained within the database and the results displayed on the screen. Messages as well as guidelines are also displayed throughout the process, providing guidance to the user. At this stage it should be pointed out that the prototype is not an attempt to develop a job-shop algorithm. Job-shop scheduling [Panurak 1991] poses a number of challenges when solving a scheduling problem. Some of them are related to the general job shop environment. For example some costs are more important than others. Different objectives, even multi-objectives, require different knowledge or problem-specific heuristics. It is impossible to manipulate all the constraints in the manufacturing systems. The real world changes unexpectedly. Since resources are used by all jobs, the assignment of one operation influences the whole schedule. It is difficult to predict what the result of assignment or reassignment of an operation will be. Operations are highly dependent on each other. Computation of complex algorithms takes too long. With n jobs and m machines, the number of schedules to be evaluated are $(n!)^m$. Every additional job will cause the computation to increase exponentially. These are some of the problems the job-shop scheduling algorithm faces and although there are various approaches to job-shop scheduling it is beyond the scope of this thesis. The prototype does not schedule or re-schedule resource allocation. This would be impossible bearing in mind that the control of these decisions is spread across the supply chain. Although the prototype can suggest allocation and re-allocation of resources the implementation of such changes lies in the hands of the parties involved. The purpose for which it was developed and intended is to provide information regarding supply and demand of goods across the supply chain. It can show when a company needs extra capacity of x or has spare capacity of y.

6.3 Additional Features

The prototype system supports a number of queries regarding the companies in the model and their data on supplies. There are a number of built-in SQL queries which produce reports in text format, as well as queries which display the results on the maps used by the system as Figure 6.12 shows. Both types of query are examined in the following section.

Figure 6.12 Map of the North East

6.3.1 Database Queries

One of the features of the prototype is the built-in queries. Currently the prototype includes 8 built-in queries. They produce reports on the companies that need extra supply, have extra supply to offer, or are waiting in the queue for a new company to enter the model. Figure 6.13 shows the list of built-in queries.

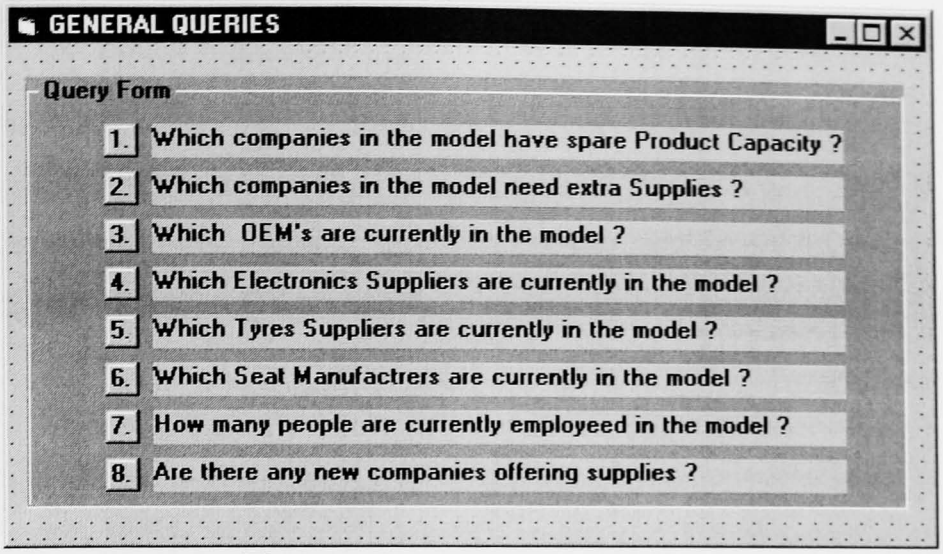

Figure 6.13 Built in Queries

The replies to these queries are based on the data stored in the database.

6.3.2 The Use of Maps

Maps allow to represent the data graphically. This option is always used in conjunction with the *'Visualize'* option. The *'Visualize'* option enables the user to select which enterprise relationships (e.g. Option: Visualize Nissan Relationships) will be displayed on the Map. The prototype currently caters for four maps. There is a general map of U.K. which is divided into three others showing in more detail parts of the U.K., mainly the:

- Northeast,
- Midlands,
- London.

When the map option is selected the appropriate map appears on the screen. Each enterprise is placed at a location on the map as declared in the database under the field name *'base'* as shown in figure 6.12.

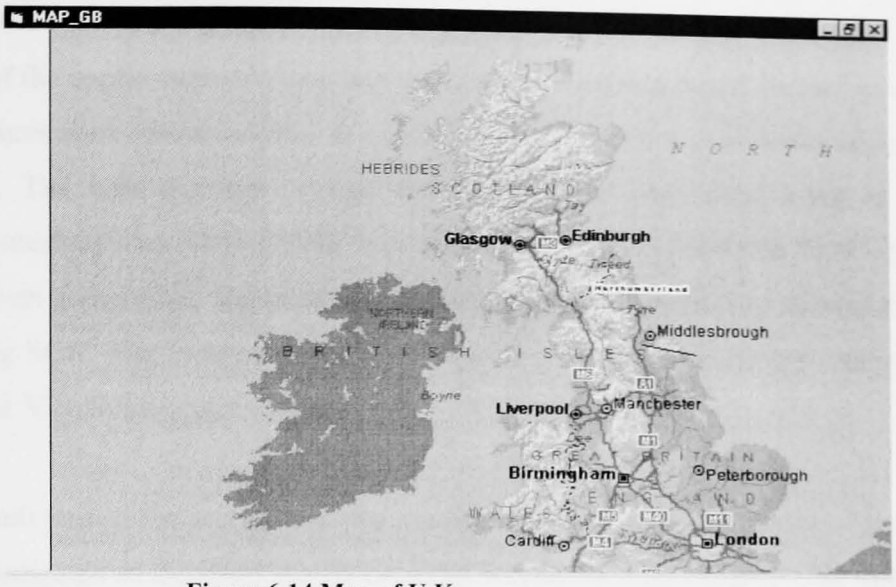

Figure 6.14 Map of U.K.

Figure 6.14 shows one of the four maps provided by the prototype. The system supports queries in the form of *'Show me the links of Vauxhall in the North East' or 'Show me the links of Rover in the Midlands'* as Figure 6.15 shows.

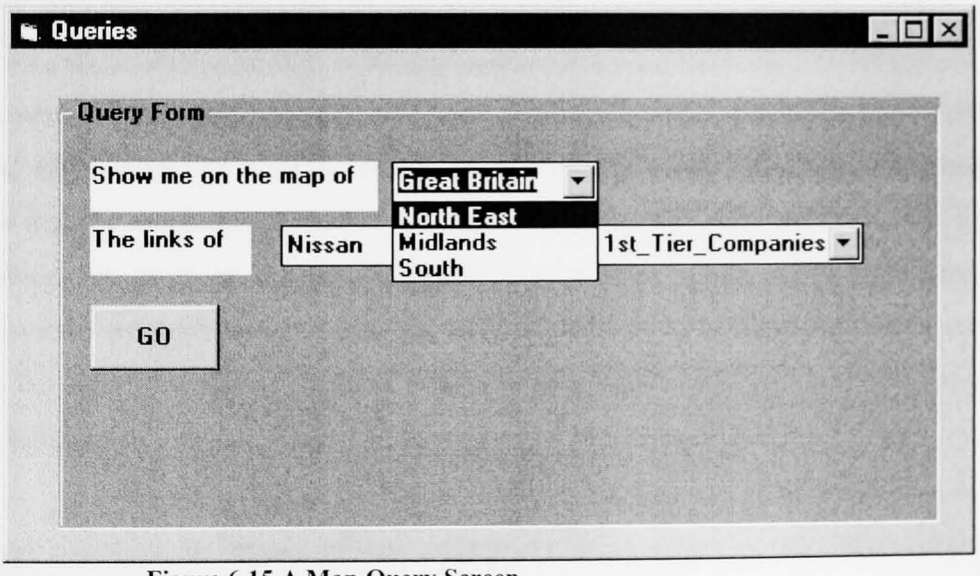

Figure 6.15 A Map Query Screen

Figure 6.15 show a query form. The user can use the combo boxes to change the parameters of the query. The first combo box shows a selection of maps. The other two boxes offer similar options regarding the integrator (e.g. Nissan as Figure 6.15 shows) and the 1st tier (level) companies.

6.4 Conclusion

The selection of the applications (Access 2.0 Visual Basic 4.0) was based on ease of use. The data structures were tested in order to check that consistency was maintain amongst the key fields. The normalization process ensured that no duplication listed in the database. The interface was developed in Visual Basic 4.0 having one main form which controls the seven forms used for handling the dialogues with users. The queries were structured using SQL. The linkage with the database was done automatically using the *.DLL* files of the Visual Basic environment.

The most difficult part of the process was the development of the data structures. Most of the time was dedicated to deciding which fields to include in order to assist the queries and produce the appropriate results.

Some problems also appeared during the testing of the queries. The problems appeared due to faults with the communication protocol that connected the interface with Access 2.0. After the files had been replaced the operation went according to the time scales.

One of the conclusions regarding this prototype is that relational databases may not be able to hold and manipulate a large number of entries, since the amount of cross links would grow and the numbering system that needs to support this structure would have to be very robust. So prior to the development of a realistic model using real data a numbering system would have to be constructed to support this structure. It therefore may be wiser to consider developing the data structures in a knowledge base enabling cross-links using inferences.

The prototype assessed the impact of new enterprises being added or deleted in relation with the first level enterprises. One can imagine that if the prototype was to be developed in a full size system and more levels of supply chain were supported then again we would need a better mechanism such as heuristics to assess the impact of change across the various levels.

Although the thesaurus structure could be generic to support other types of agents, the majority of the PAD rules that would help the model to animate, would have to be

developed specifically for each particular type. Again this would be wiser to develop in a knowledge base rather than as a set of conditional statements since the number would be large and the statements would be complex.

Although conversation theory has not been applied to the prototype it could prove to be a powerful mechanism for calculating the impact in the model whenever a change occurs. It would provide a more detailed account of how relationships can evolve when changes occur as well as show the responsibilities held by each entity in the model. The prototype was developed as part of this research work for an industrial partner. It was developed before the development of the conversation theory and therefore the theory was not included in the prototype. However the prototype helped us realise the importance of characterising relationships that were later addressed in the conversation theory.

Chapter 7 Summary & Conclusions

7.0 Introduction

In the following paragraphs we focus on the main aspects of this thesis. The reader is reminded of the definitions we gave in this thesis regarding enterprise modelling, what inspired this research and how ideas were developed. In brief we examine some of the major points of the literature review as well as the methodology developed in chapter 4. The chapter concludes with some of the ideas for further research and the contributions of this thesis to the research community.

7.1 Review of Enterprise Modelling

The most important aspects of this thesis are summarised in the following sections. During this research we recognised early on the importance and the impact of new technologies (Internet and telecommunications) upon business enterprises in the 1990's. Virtual enterprises require lower cost as there is no money spent on setting up premises and hiring staff. The Internet enables enterprises to advertise, trade and connect with potential customers without a physical presence. A survey (carried out by the Economist in 1999) showed that 90% of companies had some sort of Internet presence. In fact this new technology has totally transformed not just how companies interact and connect with customers but also their roles and responsibilities. Outsourcing is also becoming more and more common. So by reducing the need for a business to provide all the services required, it can develop numerous short-term relations cultivated with other businesses and networks of small businesses or individuals to carry out the work. This dynamic environment needs a dynamic model to assess the impact of changing roles, relations and formation of new businesses.

Chapter 2 defined the term 'enterprise modelling' and related terms, to describe why there is a need for dynamic enterprise modelling and what makes a successful enterprise model. An enterprise could be a bank, customer services, factories and so on, however we define an enterprise as "any organisation that is engaged in any type of activity i.e.

manufacturing or simple information processing". In a general enterprise modelling is describing various aspects of an enterprise in a clear and precise manner. The aim of enterprise modelling is very broad and ranges from equipment and computer systems, to human resources, manufacturing, distribution and marketing. There are a lot of different definitions for enterprise modelling depending on context and aim. [Barzdins 1997] stands out as it covers a wide range of development. It states that enterprise modelling is a new type of programming in which the executor of the program is the enterprise as a whole, not the computer.

Enterprise modelling generally serves a particular purpose and either provides a holistic perspective describing business processes, or a specific perspective concentrating on re-engineering of certain aspects of an enterprise's operations.

Later in the same chapter we reviewed various enterprise modelling definitions, that differ in content and aims. They vary from enterprise modelling as a computational representation of enterprise operations [Gruininger 1996], to enterprise modelling as a language used to describe an enterprise and permit alternative models to be considered in design, planning and so on [Fox 1996]. However enterprise modelling definitions can be broadly divided into two groups; enterprise modelling as a computational model (Gruninger 1996] or as a graphical or diagrammatic representation of all business processes, activities, roles, resources etc. [Whitman 1997]. The first type attempts to establish a common platform across an enterprise's boundaries, and tries to solve communication problems. The latter type of enterprise modelling definition relates to graphical models of processes, resources or roles mainly for the purpose of business process re-engineering.

7.2 The importance of Enterprise modelling

Due to large amounts of data used to describe supply chains or any other networks of agents, models tend to focus upon one specific area. Despite this enterprise models can help to visualise very complex situations, and thus visualise opportunities and problems. In a competitive environment, the critical steps are identifying every available

opportunity, predicting them before they occur, and taking advantage of them by taking appropriate action.

7.3 Aims of enterprise modelling

The scope of an enterprise model usually focuses on either operational, tactical or strategic level. The first level (operational) would deal with short-term forecasts within day to day transactions. The second (strategic) would deal with supply chain or network issues such as distribution strategies, sales forecasts, marketing plans etc. The third level (strategic) would deal with long term planning and production and looking into setting future objectives and strategies.

The three different levels serve different purposes due to different objectives and level of information used. Strategic level is useful -for example- for long term resource planning and such modelling would help with future objectives and assist in decision making regarding future investments and exploitation of opportunities. Tactical level modelling could assist –for instance- in making adjustments to inventories, storage, raw materials and transportation. This would help with resource planning and partially process planning and would also help the business to adapt to changing market conditions. The operational level modelling deals with specific products with the aim of i.e. determining the best time to manufacture and schedule production. It is at this level that business process re-engineering is applied.

One of the premises if this thesis is that enterprises are open systems by nature and are affected by their surrounding area. They are autonomous entities but also interconnected with the outside world. These links or dependencies are very important to enterprise modelling and play a huge role in their success or failure. A car manufacturer ensures that an x amount of cars are produced each month by ensuring that not only the businesses of the enterprise work 100% but also that the links with the outside world provide 100% input. An example of this can be found at the integrators operations in the automotive industry. The majority of them are dependent on outsourcers for such things as circuit boards, tyres, seats etc. If for example the electronics company was unable to provide

circuit boards for a period of 1 month and supposing there are no other suppliers to provide the circuit boards then car production cannot continue as normal, and the production of cars must be reduced. If such an event took place then the entire enterprise model will be effected either directly or indirectly. So malfunctions at the lowest layer of the hierarchy can have an impact all the way up to the top level. The circuit board producers will affect the electronics company which will affect the integrators job of producing x number of cars, thus the tyre company will also be affected because the integrator will demand less tyres. This may certain roles and responsibilities in order to deal with the problem. The aim of the thesis is to provide a framework that produces enterprise models that are capable of animating the roles of the different agents. This is a difficult task since enterprise models require many different parameters to be taken into account. An important factor to note is that the author has deduced that the scope of the model is proportional to its dynamics, thus the more specific a model is the more flexibility it has. Enterprise modelling can be more dynamic when it targets very different aspects such as logistics, services, ownership, resources, responsibilities and so on.

The outcome of this research is the development of a system where scenarios such "what-if" –regarding the roles- can be applied and the implicated changes and their impact upon the model can be assessed. Chapter two has provided a clear overview of different definitions of enterprise modelling, and discusses general features such as characteristics and scope of models. The author draws attention to the importance of including characteristics of openness, completeness and accuracy when modelling enterprise dependencies. During the development of an enterprise model the purpose and the level of support it can be intended for, can be assessed. The boundaries of enterprise models must be expanded in order to include a wider view, which is essential in a global market. However it should be noted that the information to be handled is undoubtedly vast.

7.4 Enterprise Modelling Literature

There are trends and fashions of different modelling tools, such as modelling and integration, CASE tools, UML etc. With this diverse number of different methods in place, users, analysts and modellers have observed an ensuing lack of consistency. Many

tools and methodologies were reviewed in chapter three. It is sufficient to say that in this chapter there will only be space to mention some and briefly describe the most influential ones. Chapter three goes into these areas in depth, and thus should be consulted for a much more in depth discussion of the following methodologies and tools. The next section discusses the importance of modelling methodologies, in order to emphasise the importance of model dynamics. Many modelling methodologies take a static approach to modelling and their outcome is usually discarded soon after development or as soon as it does not describe the current situation. On the other hand dynamic models target specific functions within an enterprise, require a lot of effort in building them and they cannot be re-used by other functions.

7.4.1 Static Methodologies

CIM-OSA, a reference architecture, was created with the intention of modelling the scope of manufacturing enterprises and its emphasis rests at three basic levels: requirements definition, design specification and implementation design.

The functional model defines what is required in terms of structure, content, behaviour and control and how this design will be implemented. The information view employs knowledge representation techniques in order to capture the semantics of information: generalisation, aggregation, particularisation and generalised relationships. The resource view retains all the relevant information about people, hardware/software and equipment, taking a hierarchical approach in retrieving information. The organisational view consists of all the relevant information relating to responsibilities in order for human decision making process.

The GRAI method provides a modelling framework and structured approach to guide the application of the methodology. It is a composite of a physical subsystem, a decisional subsystem and an informational subsystem, each dealing with different aspects of the same enterprise. P Bernus, a field leader in enterprise integration, emphasises the need for models, tools and fundamental principles in order to support the entire life cycle of an enterprise. So there is a need for a model that can adapt to a changing enterprise. PERA

(Purdue Reference Architecture) methodology defines a generic information system in terms of human based tasks and was developed to assist in modelling computer integrated manufacturing enterprises. Tasks are divided into an information stream and a manufacturing stream.

The enterprise project promotes the use of knowledge-based systems in enterprise modelling and its goals are to support organisations efficiently in the management of change. The emphasis of the project rests upon helping to manage change through the strategic use of IT and management innovation, from the approach of addressing major problems such as the impact of change, communication, process consistency amongst IT systems. It is important for businesses to increase both their relative and absolute performance due to a combination of internal and external factors. Management of change needs businesses to monitor and improve their performance against strategic objectives, in order to be successful. This process requires modelling methods to be improved and replaced by a framework that integrates methods and tools appropriate to enterprise modelling and the management of change. Furthermore enterprise modelling requires an ontology that forms the basis for the framework for integrating tools and methods. The framework supports both a general base of practical knowledge based modelling tools and methods for business application, and it has evolved alongside existing and emerging standards in open systems and knowledge representation. The Enterprise Toolset utilises an integrated framework using an agent-based architecture and is a composite of procedure builder, an agent toolkit, a task manager for integration, visualisation and an enterprise ontology. Each of these components has a different purpose and approach, yet they work together using an integrated framework.

The Toolset has been implemented in real business applications such as Lloyd's Register, IBM and Unilever. Each application has a different purpose for the evaluation of the Toolset. Lloyd's used the results for strategic planning through more effective modelling and re-engineering of business processes, while IBM in the UK used the results in remodelling of its internal organisation. Unilever used the results within R & D activities.

7.4.2 Dynamic Methodologies

The previous section discussed static enterprise modelling methodologies this following section discusses dynamic enterprise modelling. Dynamic behaviour is most well known in IDEF models, however other well–known methodologies exemplifying dynamic behaviour are WADE and TOVE.

The integrated definition method (IDEF) is a structured approach used for enterprise re-engineering and BPR [DeWitte 1998]. IDEF helps enterprises to understand processes and how they work, what input is expected and output produced. Models that are developed using an understanding of the 'as-is' of the enterprise can visualise a 'to-be' situation, assess it and implement it. This is the greatest advantage of the IDEF system as it allows an enterprise to be viewed as it is and as it can be in the future. There are numerous IDEF methods each method deals with different aspects or levels of an enterprise. IDEF methods are used in turn depending upon the stage of analysis. The IDEF method is very important to dynamic approach to enterprise modelling, and is discussed in much greater detail in chapter 3.

The WADE method is dedicated to the support of design and analysis of workflow systems in context of continually evolving business processes. The business process is designed alongside the workflow process, which is more effective than having the workflow process and the business process modelled separately. By bringing these two activities together it enables advantage to be taken to achieve business goals. WADE consists of 8 different phases, each focusing upon a different task from acquiring business process definitions to design of workflow models and finally execution of the workflow itself. WADE is advantageous because it enables planning within an enterprise and used for analysis and planning of tasks. It is also capable of identifying opportunities for BPR and evaluating alternative business process designs. It also permits process modelling and simulation to be used in parallel to design and engineer workflow systems.

GERAM on the other hand, can be described as a toolbox of concepts for analysing, designing, implementing and maintaining an enterprise. The dynamic concepts that have

been built within its methodology allow it to accommodate change within an enterprise environment. GERAM is comprised of a generic enterprise reference architecture, a generic enterprise engineering methodology and generic enterprise models. The advantages of such methodology such as GERAM is that its consistent modelling environment to support the enterprise and a model of the enterprise is developed by different approaches simultaneously.

TOVE is an ontological approach aiming at creating a common and shared terminology within an enterprise. Within TOVE the meaning of each term is defined in an easy way to understand. It describes semantics as a set of axioms and defines a set of symbols for depicting a term in graphical form. This knowledge representation can be evaluated by a set of criteria [Fox 1996] such as: generality, competence, efficiency, perspicuity, transformability, extensibility, granularity and scalability. The aim of TOVE is to represent an action that is of central importance to all enterprises.

7.4.3 Dynamic modelling tools

This section deals with dynamic modelling tools including the multi-agent approach to modelling supply chain dynamics, the aim of which is to achieve supply chain optimisation by implementing different scenarios. This approach is considered as dynamic because supply chains are considered multi-agent environments and involves business entities working together closely and being interdependent.

The NIMBUS model is a software package created for dynamic process modelling. Modelling and simulation processes are differentiated and this enables users to concentrate on the modelling task. NIMBUS contains a set of computer programmes that enable modelling and simulation of processes in the form of algebraic equations. Details about both of the above methods can be found in chapter 3.

Later on chapter 3 discusses organisational modelling commercial products, which deal with primarily business process re-engineering. These tools include the Proforma tool that provides tools and guidelines for business software development. These tool and

guidelines are for the purpose of BPR, business object analysis, client server design and development with the aim of integrating information and business technology. Other commercial products are Software Data Environment (S.D.E.) which is focused upon implementing 'what-if' scenarios and Panagro that takes the approach that an organisational model is the capture of the mental recreation of an organisation. The Pangaro tool and its modes focus in the achievement of goals of the organisation, the activities of its members and responsibilities of people involved [Panagro 1999].

This overview has briefly mentioned the main parts of each method covered. The author discusses in much depth all of the methodologies mentioned here in chapter 3. Although each method and tool in this section has a different approach to modelling, they emphasised dynamic concepts that focus on an enterprise's interaction with external factors as well as internal ones. The importance of these enterprise modelling methods is that the scope of a model is proportionate to its dynamics. Static models often model a widespread area whilst dynamic or models tend to specify one particular area. Often a dynamic model will have to be created for different levels of an enterprise and in that way a whole enterprise can be dynamically modelled. The various methods mentioned provide support for different types of integration. TOVE for example provides a structured approach to enterprise integration while CIM-OSA requires much more detailed information in order to accurately model an enterprise.

The author raises an important distinction between static and dynamic models, the former allows the picture of the enterprise as perceived at the time the model was created while the latter allows changes to be accommodated and reflected in the picture of the enterprise. Our EM is focused upon how different components within an enterprise communicate as well as how enterprises interact with and relate with one another. This communication both internal and external is crucial to an enterprise's success or otherwise.

7.5 A Dynamic Approach to Enterprise modelling

Chapter 4 points out the problems with the current methodologies and stresses the need for modelling environments as open systems. Since we're modelling systems using open systems architecture it is wise to maintain this architecture when it comes to modelling. Many of the methodologies examined were developed with a computer-integrated environment in mind. They emphasise in developing different views of an enterprise although they often omit the fact that enterprises are open systems that are affected by changes in the outside world. An account of open systems architectures as well as the main principles of open systems is given in the chapter. Many of the ideas used in chapter 4 to add dynamic concepts to modelling were borrowed from conversation theory and AI literature.

It was early on conceived that prior to any type of animation we needed to build the initial state or picture. In order to draw the initial picture of the model and at the same time maintain the hierarchies of links and agents we used the enterprise thesaurus. The thesaurus is a type of data structure that was proposed by this thesis for storing information about agents while at the same time keeping track of hierarchies of links. Information about agents was stored along with a series of predicates such as 'uses' or 'supplies-to' or 'supplied-by'. This information allowed the initial picture of the model to be drawn. The dynamic concepts of the approach were developed using conversation theory and AI paradigms.

Conversation theory is a framework for categorising relationships between agents. Relationships are evaluated against four variables such as significance, mutuality, control and capability. The variable of significance shows how the benefits of the relationships are distributed. If for example a relationship between two agents is of symmetric significance then we can conclude that both agents share the same benefit. If however the significance is asymmetric then one of the parties benefits more. Mutuality on the other hand shows how responsible each party is for the benefits of the other. A relationship with symmetric mutuality implies a partnership or co-operation whereas asymmetric mutuality would imply competition. If this variable is combined with significance then

various combinations can be made as shown in table 4.1 and 4.2 of chapter 4. The third variable, that is control, shows which of the parties has control over the creation or the break up of the relationship. Capability -the fourth variable- shows whether the two parties share resources during their relationship. These four variables give great flexibility to the approach defined in chapter 4 in terms of categorising relationships.

We used the same data structures to define the domain of the model, store information about the current situation as well as express a change in the current situation. The domain that determines the nature of the model as well as the types of relationship that can be created is expressed in PAD rules. PAD which stands for preconditions, add-list, delete-list, is a series of conditions that show how relationships can change and the effect they have on the entire model. PAD rules change according to the model although in chapter 4 we presented some generic facts about them that allow mechanical generation of some of those. Although in chapter 4 rules are presented as a series of conditional statements we acknowledge the complexity of producing these data structures and we explore knowledge representation techniques for doing so. These findings are presented in chapter 5. The movement of the model is restricted by a series of lisp type lists that are called premises and restrictions. These types of list are generated along with the initial model (initial situation) and their aim is to prohibit the model from developing relationships between its components that we consider invalid. Invalid relationships can be considered these relationships that are unrealistic to be found in a real world situation. For example a competitive relationship between two services or manufacturers of the same product cannot be of symmetric mutuality. Having explained the basic elements of the approach we carried out the following steps.

- Description of agents in the enterprise thesaurus
- Generation of states using the thesaurus descriptions expressed in lisp
- Generation of PAD rules
- Generation of premises and restrictions expressed in lisp

One may ask 'how do you assign conversation theory values from the thesaurus initial state'. By examining models presented in chapter 4 (health care, brokerage, automotive) we concluded that although those models did differ in terms of relationship structures roles, objectives etc, there was a number of similarities between them. Those similarities can be used for developing the initial picture from the thesaurus as well as some of the PAD rules that define the domain. The rules of the thesaurus are:

- Control is always assigned to agents with higher authority.
- The level of authority is assigned according to the level in the hierarchy
- Agents that belong to the same structure are assigned equal control and positive capability
- Agents that are inter-linked but they do not belong in the same structure are assigned symmetric significance but asymmetric mutuality. Control is passed to the agent with the larger structure.
- Capability between agents of different structures or sub structures is always negative.
- Agents under the same structure are assigned symmetric significance and mutuality

Based on these rules, knowledge can be represented by Lisp type sequences in order to describe a particular situation *s*. Some of the PAD rules can also be generated automatically according to the agent descriptions of the enterprise thesaurus. While the enterprise thesaurus describes and evaluates the actual links, PAD rules describe how these links can evolve. The rules for generating PAD rules are the following.

- Relationships between agents of initial value s[on] m[off] can be transformed to s[on] m[on] or s[off] m[off] or s[off] m[off]. Control is passed to the agent with higher authority while capability is positive only when mutuality is on.
- Relationships of initial value s[off] m[off] cannot be transformed. Control remains equal while capability is positive.
- Relationships of initial value s[on] m[on] must have positive capability. They can be transformed into s[off] m[on] with positive capability and control passed to the agent

with higher authority. If the values turn s[off] m[off] then capability becomes negative while control remains as it was.

- If capability turns to negative then the relationship values turn m[off].
- Capability cannot turn negative for a relationship of s[off] m[off] value.

The rules have been derived from the theory of conversations as described in chapter 4 and tested against 3 case studies. All case studies led to the same conclusions. According to these rules we can generate a list of premises and restrictions. Evaluation of each state is achieved by carrying out the following tasks:

- Validation of action against restriction
- Authorisation of action from premises
- Establishment of new state from the PAD rules
- Generation of new state
- Generation new premises
- Generation of new restrictions

These rules do not contradict the knowledge representation techniques we reviewed in chapter 5 for generating such as a domain. They are a series of similarities that were found in all case studies examined during this research. The specifics of each model make it necessary to consider their domains separately. Bear in mind that all these steps are lists that are being processed using the content each one of them. Processes are triggered by actions. Action lists specify a goal state. The data of the state description list (premises, state, restrictions) is later decoded to present a new state. The following paragraph highlights the conclusions that were made during this research as well as the contribution of this thesis to the research community.

7.6 Conclusions

Since the early stages of the research it became apparent that many authors, researchers or developers used the term 'enterprise modelling' to refer to methods and tools that had

little or nothing common. For example, case tools that assist business process re-engineering differ from construction of ontologies. Both processes however were described by the same term. Therefore the thesis distinguished between modelling or diagrammatic representations of networks of agents which can take the form of enterprise components and communication platforms that promote information exchange.

Within the context of enterprise representation (diagrammatic or mathematical) we distinguished between structure oriented, process oriented and resource oriented models. Each of these types was further categorised to descriptive, prescriptive, constitutive and operative models, all of which have different aims and purposes, demonstrate different aspects and are based on different perspectives.

The main reason for enterprise modelling is that the structure of enterprises and the algorithms that describe them have become very complicated. These issues plus incompatibility led to the need for reengineering in order to keep up with technological updates, and the need for a formalised procedure for developing schematics of an enterprise's processes. Enterprise models help to anticipate the effect of different conditions on an enterprise and in order to minimise risk they can make forecasts and experiment with fictional scenarios. A great benefit of using enterprise models is the ability to 'see' opportunities and prioritise actions. With a suitable enterprise model the amount of change and the effect of the changing strategies on the model can be predicted. By creating fictional scenarios we enable possible problems and opportunities to emerge and this is a great advantage in a competitive market. Relationships in a supply chain for example can be modelled and the consequences of a relationship being deleted or added can be assessed for possible modes of action.

One interesting analogy is the scope of the enterprise model versus its dynamics. These two aspects are inter-dependent and inter-related. An enterprise model can be expected to keep production costs low, improve product quality, reduce time scales, or if in the service industry identify customer needs, perform promotional activities and cut down on service fees. On top of this there are constant challenges which an enterprise will face

such as an integrated working environment as opposed to autonomous departmental units. Therefore a model must represent not only individual units but also relationships and dependencies of these units. In order to assess these challenges we need models build with dynamic concepts. The analogy which we identified in this thesis is that the wider the scope of the model the more static it becomes. On the contrary the more specific the target the more dynamic the model. This is why mathematical models work well with small domains (sales, distribution) where the relationship between inputs and outputs can be expressed in a formula, but fail to consider a wider picture of the enterprise. The 'big picture' is therefore usually represented using diagrammatic schemas.

There are three main aims of enterprise modelling and these are; to be able to assess change and it's implications, to specify the scope that the model is dealing with and to measure performance. There are different types of enterprise models, yet they all should embody certain qualities or characteristics; an enterprise model should be able to assess changes that might occur in an enterprise network and their impact on an enterprise's operations. A successful enterprise model must be able to monitor market signals across a network and make consistent forecasts, regarding such things as customer demand ordering patterns and restocking algorithms. To be successful an enterprise model needs a number of different factors to ensure this such as; completeness, wide scope, dynamics, expressive power and the quality of openness. The last quality is the one that is most difficult to incorporate within an enterprise model. An enterprise model needs to be open in order to avoid becoming static and therefore discarded as several surveys have proved.

Dynamic modelling is essential since static enterprise models fail to represent accurately the evolving state of an enterprise as a part of a wider market or supply chain due to the vast amount of changes in business roles, responsibilities, technological achievements and so on. The scope of an enterprise model is ultimately dependent on how dynamic such a model is. Models that cover a large area tend to be more static than models that focus on a smaller more specific area.

In the light of this analogy -and in a sense classification- chapter 3 discussed the different types of modelling methodologies and products associated with E.M. We concluded that there is a distinguishing line between architectures, tools and methods associated with enterprise modelling each of which targets different aspects of the enterprise. The basic type of E.M defined as static considers a wider view of the enterprise, whilst dynamic models focus on a more specific aspect of an enterprise such as sales.

Many authors when it comes to modelling state that the aim of a modelling process is to create an abstraction of the world, and reflect characteristics that exist in the real world, to a chosen level of degree. This specificity allows humans to understand complexity through abstraction as well as understand the pieces of abstraction. It is part of human rationality to break things down into manageable chunks and this is something that can be observed during enterprise modelling too. Modelling methodologies offer alternative views of the enterprise (information view, physical view) as well as alternative aspects (responsibilities, processes). Similar to the way we decompose systems some methods divide enterprise models into different views. Some of them go a step further by identifying the connection and interaction between those views.

CIM-OSA is an open systems architecture, and can provide a number of views to support all phases of the modelling cycle. The CIM-OSA method, combines functional decomposition and utilises a three level approach composed of function, activity and information modelling. ICAM has developed an architecture using tools/views such as IDEF0 and IDEF1. The IDEF method models information, activities and data requirements of each function as well as changes which occur over time. NBS architecture utilises hierarchical control composed of five levels, with each level being a composite of further sub-systems. Impact architectures attempt to bridge the gap between different software that enables global planning and production control strategic planning in real time. SSADM allows different perspectives to be taken of a system. The Structured Analysis and Design technique uses a top-down approach to model the structure of a system. The final and most relevant approach to modelling for this thesis is the Object-Oriented approach as it consists of two phases; an analysis phase, decomposes

component functions of manufacturing in order to trace information flow in the infrastructure, and a design phase. One could conclude from this paragraph or by reading in detail about these methods that the enterprise models they develop are in fact series of representations each one targeting a particular aspect of the enterprise. The exact same principle of decomposition has been widely applied in systems analysis and design. There are also a variety of modelling tools available such as the IDEF modelling tools, structured systems analysis, GRAI grids and nets (model decision-making process). Generally speaking these modelling methods provide a basic outline of how modelling tools can be integrated to model a specific system. It should be stressed at this point that many of these modelling methodologies were build for BPR or I.S. engineering and tested on CIM enterprise systems.

Business process re-engineering is the process of identifying all the elements of an enterprise's functions from the perspective of the information required, labour and equipment and so on. This term is often utilised within the field of enterprise modelling, and its aim is to assess business functions to see whether each function is producing the desired effect and if there is any room for change.

Enterprise I.S. Engineering is a branch of requirements engineering dealing with early stage of system design to promote understanding, use and development of technology in all areas of possible applications. Its purpose, like other enterprise modelling methodologies, is to identify defects with current architectures and to introduce or suggest alternative designs. [Gustas 1998], defines five stages of enterprise engineering: systems analysis of the enterprise current situation (as-is), enterprise modelling and integration, enterprise change analysis, enterprise business processing engineering and enterprise assessment exercise. The advantage of this system rests in the fact that several aspects of the information system are being recorded and are directly applicable to the development of the desired information system.

Communication, which is another area that was targeted by enterprise modelling was tackled with the development of ontologies. Ontology is a new term in the context of

E.M. and can be defined as a specification of a conceptualisation [Gruber1996]. Ontologies are essential for knowledge sharing and re-use. In a nutshell ontologies are large dictionaries of data definitions as well as descriptions of their functionality and use. Often computer systems enable different departments to use a single knowledge base with each department being able to interpret information very differently, and use this information for very different purposes. The aim of an ontology within an enterprise is to support integration within the boundaries of the enterprise via a common knowledge store. It also increases the communication potential and avoids possible confrontation. Projects such as the 'Enterprise Ontology' are examples of these principles in practice.

However the problem we tackled in this thesis was not communication or representation but rather that of animation. The approach we took was to combine several paradigms in order to build some dynamic concepts within our method and therefore allow models to animate. Conversation theory is combined with some principles of AI in order to represent and animate models. During that time we made an interesting conclusion about dynamic modelling. The ways we represent problems in AI and construct knowledge structures about facts and beliefs as well as the ways we explore possibilities can be used to add dynamics to models. In chapter 4 we show this by using the example of a robot living in a house of four rooms and his job is to move and carry around furniture. We pointed out that the way we construct the situation in data structures to represent the house with its boundaries and its furniture can be used to modelling and more specifically for enabling animation. We compared the robot's house with an enterprise and found many similarities. The rooms can then be considered to be the components or departments of an enterprise and the furniture the dynamic elements such as roles or responsibilities that can be shifted around. This parallelism allowed us to practice and experiment with a number of principles based on AI literature. This process led to the construction of the enterprise thesaurus, knowledge representation using PAD rules and animation by processing lisp type data structures.

7.7 Further research

Chapter 4 raises some questions regarding the methods that need to be employed in technical terms for constructing such as methodology. We said earlier on that many of the ideas of the model were based on AI literature. Chapter 5 explores AI literature and gives answers to questions regarding the methods employed for making multiple movements, finding goal states as well as constructing a model domain.

It can be the case where an action list calls for a goal state that requires the model to animate more than one time in order to reach that state. In order to do this we need to know all the possible movements of the model i.e. all states, and outline a method that will evaluate every state against the goal. The method for planning and finding plans is called heuristics. Heuristics can be a simple evaluation process or very complex depending on the amount of possibilities that need to be evaluated. The main aim of AI heuristic searches is to decrease searching time as it can be considerable in some systems i.e. a game of chess.

Chapter 5 explains how a tree of possibilities can be developed out of the PAD rules. There are basically two ways of generating and evaluating possibility trees. These are depth first search and breadth first search. The first would take one agent and explore its possible moves to the maximum whereas the latter would explore all agents one level at a time. Both searching methods come with a set of advantages and disadvantages as it is explained in chapter 5.

The author goes into detail about AI heuristics and the research that has been done in the area. Some of the heuristic methods presented and examples may not be directly related to enterprise modelling. They are however presented as a means for further research that could assist or eventually become related with the field of enterprise modelling.

Chapter 5 also points out the difficulty and the complexity of the task of constructing the data structures for storing PAD rules. Knowledge engineering is the area that deals with the area of developing data structures that are not related under a numbering system

(relative databases) but by inferences and deductions. We have identified 5 different levels of representation for this task. The implementation level deals with developing the data structures for storing PAD rules. The logical level should provide sufficient logical properties to represent knowledge. The epistemological level deals with types of knowledge structuring primitives that are needed. The focus is at the development of a set of primitives that would enable the description of world objects as formal structural units as well as their interrelationships. The ontological level is concerned with meaning since knowledge representations are sets of ontological commitments. Primitives represent an agreed meaning between users. Finally the conceptual level is concerned with primitives to be included in a knowledge representation language. Later on in the same section we provide a review of the basic representation structures such as frame formalisms and rule formalisms. The chapter concludes with an account into the software tools that are widely known and used for constructing knowledge bases and knowledge based systems. A number of companies are currently developing organisational models in the form of ontologies, knowledge bases or business models. Inference [eGain 2001] IBM [King 1995] and Loop [Loop 2001] provide such solutions which aim to produce formal representations of the two conceptual components namely the *entity* and the *link*. Depending on the domain they address a number of issues such as common terminology, roles and responsibilities, ownership and others.

7.8 Contributions

This thesis presents a methodology for developing and animating enterprise models. It can serve as a tool in enterprise modelling as well as enterprise engineering or re-engineering. The strength of the methodology is that it allows new roles, responsibilities and relationships to be exploited, tested and evaluated. It provides the means for assessing the impact of change without being committed to change.

The development of the dynamic concepts for enterprise models was the major contribution of this research. By providing the means for evaluating and classifying enterprise relationships we move a step towards automatic generation of models and movement based on AI heuristics and reasoning. The research into the area of AI, the

parallelism that were made and the similarities that were found has opened new directions for researchers and academics to exploit.

Another benefit of this research is a more clear understanding of the enterprise modelling area. We showed how various enterprise models are built as well as the purpose they serve. The classification, critic and comparisons of various models will help modellers in selecting the appropriate modelling method. Finally the description of tools and technical specifications provides a realistic account of the complexity of the task of developing enterprise modelling tools and algorithms.

Chapter 8 References and Bibliography

8.1 References

[Allen 1991] Allen, J.F. *Planning as Temporal Reasoning* In Proc., 2nd Principles of Knowledge Representation and Reasoning, Morgan Kaufmann, 1991.

[Allen 1992] Allen, J.F. et al. *Reasoning About Plans*, Morgan Kaufmann, 1992.

[Anderson 1996] David L. Anderson, Frank E. Britt, Dovavan J. Favre. The seven Principles to Supply Chain Management

[Artzen 1997] Bruce C. Artzen, Gerald Brown, Terry P. Harrinson, Linda L. Trafton 1997. Global Supply Chain Management at Digital Equipment Corporation

[Barzdins 1997] Bardins J., Kalnis A. *Enterprise Modelling and Business Process Reengineering*, International Conference and Exhibition, Riga, April 2-5, 1997, http://www.gradetools.com/grade40/whitepap.htm

[Bernus 1996] Bernus P., Laszlo Nemes, Theodore J. Williams, *Architectures for enterprise integration,* pp.127-161 Chapman & Hall, 1996

[Bernus 1998] Bernus P., Uppington G., *Assessing the necessity of enterprise change: pre-feasibility and feasibility studies in enterprise integration*, International journal computer integrated manufacturing, Vol. 11,No. 5, pp430-447, 1988

[Bertalanffy 1969] Bertalanffy van Ludwig, *General System Theory: Foundations, Development, Applications,* George Braziller, 1969

[Brandimarte 1995] Brandimarte P.,Cantamessa M., *Methodologies for designing CIM systems: A critique* Computers in Industry 1995, 25: 281-293.

[Breitman 1987] Breitman R.L., Lucas M.J., *PLANETS: A modelling sysmte for business planning,* Interfaces, 17, Jan.-Feb., pp94-107

[Bussler 1994] Bussler C., Jablonski S., *An approach to integrate workflow modelling and organisation modelling in an enterprise*, IEEE Proceedings of 3rd workshop on Enabling Technologies: Infrastructure for collaborative Enterprises, 81-85. Morgantown, WV:IEEE Computer Press Society

[Butler 1999] Butler K., Clement M., Snell Q., *A performance broker for CORBA*, Proceedings. The Eighth International Symposium on High Performance Distributed Computing, IEEE Computer Society Los Alamitos, CA, USA, 1999, pp. p.19-26.

[Chen 1976] Chen, P.P.S., *The entity-relationship model towards a unified viewof data,* ACM Transactions on Database Systems Journal 1,pp9-36, 1976

[CIMOSA 1994] *Open system architecture for CIM, Technical Base-line, Version 3.0,* CIMOSA association (e.v.) October 1994

[Clancey 1993] Clancey W. J., The Knowledge Level Reinterpreted; Modelling Socio-technical Systems, *International Journal of Intelligent Systems,* Vol. 8, pp. 33-49, 1993.

[Cohen 1985] Cohen M.A., Lee H.L., *Manufacturing strategy concepts and methods,* in Kleindorfer P.R. ed., The management of productivity and technology in manufacturing, pp153-188

[Cohen 1989] Cohen M.A., Lee H.L., *Resource deployment analysis of global manuacturing and distribution networks,* Journal of manufacturing and operations management, pp81-104, 1988

[Davis 1993] Davis R., Shrobe H., Szolovits P., What is a Knowledge Representation?, *AI Magazine*, Spring, 1993

[DeWitte 1998] Paula S. deWitte, Chris Pourteau, IDEF enterprise engineering methodologies support simulation, Knowledge based system Inc, http://www.idef.com/articles/inside-the-process/Inside-the-process.html

[Dobson 1997] Theory of Conversations, Martin & Dobson, Centre of Software reliability, Newcastle university, 1997

[Doumeinghts 1995] Doumeinghts G, Marcott F., Rojas H, *GRAI approach: A methodology for reengineering the manufacturing enterprise,* Reengineering the enterprise, Brown J., O'Sullivan D. (Eds), London UK: Chapman & Hall

[Duse 1992] Duse, M.,Gharpure J, Bhuskute H., Kamath M., Pratt D.B., Mize J. H., Tool independent model representation. *Proceedings of the 2nd Industrial Engineering Research Conference,* 700-704, Institute of Industrial Engineers, Norcross, GA, 1992

[Economist 1999] The Economist, *When companies connect*, pp 19-21, June 26 1999

[eGain 2001] www.inference.com This is the web site of a provider of software and services for customer relationship management and e-commerce.

[Erdmann 1997] Sebastian Erdmann & Jan Wortmann, Enterprise Modelling with FUNSOFT nets, ISBN 0-8186-8031-8/97 1997

[Evans 1993] Evans G.N., Naim m.M., Towill, D.R., *Assessing the impact of information systems on dynamic supply chain performace,* Logistics information management, Vol. 6, No. 4, pp 15-25

[Feldner 1999] Feldner Bustin & Associates, Enterprise Modelling for Strategic Management Consultants, http://www.feldner.com/modeling.htm 1999

[Fillmore 1968] Fillmore C, *The case for case*, pp1-88, Universals of linguistic theory. Bach and Harms, Holt, New york,

[Fox 1996] Fox S. Mark, *Issues in Enterprise Modeling*, Department of Industrial Engineering, University of Toronto. 1996

[Fox et al 1996] Fox M., Chionglo J.F., Fadel G. F. 1996. A common sense model of the enterprise. Department of Industrial engineering Toronto

[Fox 1992] John Fox, Subrata Das, *Safe and Sound*, MIT press 1992

[Ganesham 1995] Ganesham Ram, Terry P. Harrinson, *An Introduction to Supply Chain ManagementDepartment of Management Science and Information Systems*. Penn State University, 1995

[Gay 1991] Gay R.K.L., Lim R., *Using IDEF methodology for functional, information and dynamic modelling of a FMS,* Proceedings of the International Conference on Computer Integrated Manufacturing. ICCIM '91. Manufacturing Enterprises of the 21st Century. World Scientific, Singapore, 1991. pp. p.97-100.

[GRAI 1993] Doumeinghts G, Marcott F., Rojas H, *GRAI approach: A methodology for reengineering the manufacturing enterprise,* Reengineering the enterprise, Brown J., O'Sullivan D. (Eds), London UK: Chapman & Hall

[Gruber 1991] Gruber Thomas, *The Role of Common Ontology in Achieving Sharable, Reusable Knowledge Bases,* Principles of Knowledge Representation and Reasoning: Proceedings of the 2nd International Conference, San Mateo, 1991

[Gruber 1992] Gruber Thomas, *Ontolingua : A Mechanism to Support Portable Ontologies*, Knowledge Systems Laboratory, Stanford University 1992

[Gruber 1993] Gruber Thomas, *Towards Principles for the Design of Ontologies Used for Knowledge Engineering,* Stanford Knowledge Systems Laboratory August 1993

[Gruber 1996] Gruber Thomas, *A Translation Approach to Portable Ontology Specifications*, Stanford University, California, Knowledge Acquisition Journal 1996

[Gruninger 1994] Gruninger Michael, Mark S. Fox, *The Design and Evaluation of Ontologies for Enterprise Engineering*, Department of Industrial Engineering, University of Toronto, May 30, 1994

[Gruninger et al 1994] Gruninger M., Fox M.,1994. Enterprise modeling. Department of Industrial engineering Toronto

[Gruninger 1996] Gruniger M, Pinto A.J. 1996. A theory of complex actions for enterprise modeling. Department of Industrial engineering Toronto

[Gruninger et al 1996] Gruninger M., Fox M. 1996. The logic of enterprise modeling. Department of Industrial engineering Toronto

[Gustas 1998] Enterprise Engineering 1997, http://www.cs.hks.se/~gustas/ENTENG.html

[Guuerino 1993] Guerino Nicola, *The Ontological Level*, 16th Symposium in knowledge engineering , Kirchberg, Austria 1993

[Hammer 1993] Hammer M., Champy J. *Reengineering the Corporation.* Harper Business, 1993

[Hammer 1998] Hammer T., Huffman L., *Automated requirements management-beware HOW you use tools: an experience report,* Proceedings Third International Conference on Requirement Engineering. IEEE Computer Society, Los Alamitos, CA. USA, 1998. pp. p.34-40.

[Heidorn 1972] Heidorn G. E., *Natural Language Inputs to a Simulation Programming System*, NPS-55HD72101a, Naval Postgraduate School, Monterey, California

[Heim 1994] Heim J. A., *Integrating distributed models: the architecture of ENVISION* International Journal of Computer Integrated Manufacturing, 1994 7.1:47-64.

[Holzmann 1997] Holzmann Gerald, *The model checker SPIN,* IEEE transactions on software engineering, Vol. 23, No. 5, May 1997

[IEMC 1999] IEMC99 Proceedings of the International Enterprise Modelling Conference, Norway, Verdal 16-19 June 1999

[Intertrans 1998] Internans home page, http://www.itls.com/ventprod.htm

[Jorysz 1990] Jorysz H. R., Vernadat F.B., *CIM-OSA Part 1: total enterprise modelling and function view,* International Journal of computer integrated manufacturing, 1990, Vol. 3, Nos. 3 and 4, pp 144-156

[Kanellis 1999] Panagiotis Kanellis & Ray J. Paul, Towards an epistemological framework for measuring the Fit of Information systems under perpetual change, http://hsb.baylor.edu/ramsower/acis/papers/kanellis.htm

[Keller 1998] Keller A. Assessment and optimisation of the thermal safety by means of dynamic modelling, Dissertation Nr. 12606, http://itcmail.ethz.ch/~hungerb/keller/ab_diss.html

[Kim 1993] Kim H., Fox M.S., *formal Models of quality and ISO9000 compliance: An information systems approach*, Proceedings of the 48th Annual Quality Congress. Milwaukee WI: an american Society for Quality Control, pp17-23, 1993

[Kim 1993] Kim C.,Kim K., Choi I, *An object oriented information modeling methodology for manufacturing information systems*, Computers and Industrial Engineering, 1993 24.3:337-353.

[King 1995] King Martin, *Knowledge reuse in business domains experience with IBM's BSDM,* Technical Report, Artificial Intelligence Application Institute, 1995 www.aiai.ed.ac.uk

[Klittich 1990] Klittich M., CIMOSA Part 3: CIMOSA Integrating Infrastructure-the operational basis for integrated manufacturing systems, International journal of Computer Integrated manufacturing, 1990, Vol 3, Nos 3 and 4, pp 168-180

[Koshikar 1994] Kochikar V. P., Narendran T. T., *On using abstract models for analysis of flexible manufacturing systems,* International Journal of Production Research, 1994 32.10:2303-2322.

[Liles 1996] Liles H. Donald, *The enterprise engineering discipline*, Automation and Robotics research institute, http://arriwww.uta.edu/eif/ent_eng.htm

[Little 1991] Little J. D. C., *Are there 'laws' of manufacturing?* Manufacturing Systems. Washington D. C.: National Academy Press, 1991

[Logica 1992] Logica News, *Logica leads team to develop tool for enterprise modelling,* December 1992, http://www.lgica.com/news/press/pr127.html

[Loop 2001] www.loop.es Commercial web site for business modelling software.

[Manna 1995] Z. Manna, *STeP: The Stanford Temporal Prover*, In Peter D. Mosses, Mogens Nielsen, and Michael I. Schwartzbach, editors, TAPSOFT'95: Theory and Practice of Software Development, volume 915 of Lecture Notes in Computer Science, pages 793-794, Springer-Verlag 1995.

[Mayer 1998] Richard J. Mayer A framework and a Suite of methods for BPR, Texas A&M University, 1998, http://www.idef.com/articles/framework/framework_pt2.htm

[Morgan 1987] Morgan Gareth, *Images of organisation*, SAGE Publications, 1987

[Morris 1993] Morris D., Brandon J., *Re-engineering your business*, New York, NY:Mcgraw-Hill

[Mujtaba 1994] Mujtaba M. Sahid, *Simulation modelling of a manufacturing enterprise with complex material, information and control flows*, International Journal computer integrated manufacturing, Vol. 7, No 1, pp29-46, 1994

[Newel 1982] Newell A., The Knowledge Level, *Artificial Intelligence*, Vol 18. No. 1, 1982,pp87-127

[Newel 1993] Newell, A., Reflcections on the Knowledge Level", *Artifical Intelligence*, Vol59,1993,pp31-38.

[Ngwenyama 1994] Ngwenyama O. K., Grant D. A., *Enterprise modelling for CIM information systems architectures: An object oriented approach*, Computers & Industrial Engineering, 1991 26.2: 279-293

[O'Sullivan 1994] O'Sullivan D., *Manufacturing system redesign: Creating the integrated manufacturing environment*, New Jersey: Prentice Hall, 1994

[Pandya 1995] Pandya K. V., *Review of modelling techniques and tools for decision making in manufacturing management,* In IEEE Proceedings on Science, Measurement, and Technology, 1995, 142.5: 371-377

[Pangaro 1999] http://www.pangaro.com/PI-Brochure/PI-Brochure.html

[Panurak 1991] H. Van Dyke Parunak. *Characterizing the manufacturing scheduling problem*, Journal of Manufacturing Systems, 10(3):241-259, 1991.

[Parsons 1998] Parsons Simon, Sierra, C. and Jennings, N. R., *Agents that negotiate and reason by arguing,* Journal of Logic and Computation, **8**(3), 261-292, 1998

[Perakath 1998] Perakath C. Benjamin, Charles Marshal, Richard J. Mayer A Workflow Analysis and Design Environment (WADE) 1998, Knowledge Based Systems Inc, http://www.idef.com/articles/wade/wade.html

[Pratt 1994] Artificial Inteligence, Ian Pratt, 1994, Macmillian press

[Proforma 1999] http://www.proformacorp.com/

[Quillian 1966] Quillian M. R. *Semantic memory,* Report AFCRL-66-189. Bolt Beranek & Newman, Cambridge, Massachusetts.

[Rumbaugh 1991] Rumbaugh J, Blaha M Object-oriented modeling and designs *Prentice Hall,1991*

[Sadeh 1994] Sadeh E. Norman, *Micro-Opportunistic Scheduling: The Micro-Boss Factory Scheduler*, The Robotics Institute Technical Report, 1994 (to appear in Intelligent Scheduling edited by M. Zweden, M. Fox, Morgan Kauffman, 1994)

[Sadeh 1996] Sadeh M. Normam, Stephen F. Smith, *Knowledge Based Supply Chain Management: An overview of Ontgoing Research at Carnegie Mellon University, Production Planning and Control Laboratory*, Center for Integrated Manufacturing Decision Systems, 1996

[Savolainen 1995] Savolainen T. D., Beeckmann P., Jagdev H., *Positioning of modelling approaches, methods and tools,* Computers in Industry ,1995, 25: 255-262

[Scott 1995] J. L. Scott, A transportation model, its development and application to a ship scheduling problem, Asia-Pacific journal of operational research 12 (1995) pp111-128

[SDE 1999] http://www.adrm.com/software.htm

[Seeley 1999] Doug Seeley & James R. Warren, Trusted linkages and systems chaos: The case for dynamic modelling of organisations, http://www.interdyne.com.au/5a58359.htm

[Shlaer 1988] Shlaer S, Mellor S J (1988) Object-oriented systems analysis: modeling the world in data *Yourdon Press,New Jersey*

[Sierra 1999] Sierra, C., Parsons, S. and Jennings, N. R. *Agents argumentatius* Proceedings of the 1st Catalan Conference on Artificial Intelligence, Tarragona, 1999

[Spitzer 1993] Spitzer R.E., *TQM:The only source of sustainable competitive advantage,* Quality Progress, pp59-64, 1993

[Swaminathan 1997] Swaminathan J., Smith S.F., Sadeh M.N., *Modelling Supply chain dynamics: A multi-agent approach*, Decision Sciences Journal, April 1997

[Szengheo 1999] O. Szengheo & B. Anderson, *Modelling the extended enterprise: A comparison of different modelling approaches*, IEMC99 Proceedings of the International Enterprise Modelling Conference, Norway, Verdal 16-19 June 1999

[Tavasszy 1997] Lorant A. Tavasszy, Ben Smeenk, Cees J. Ruijgrok, *A DSS for modelling logistic chains in freight transport policy analysis*, 7th International Special Conference of IFORS: "Information systems in logistics and transportation" Gothenburg, Sweden 16-18 June 1997

[Texas Inc 1999] http://www.ie.utoronto.ca/EIL/tool/list/bdf.html

[Top-ix 1999] http://www.metabpr.com/loop.htm

[Tham 1995] Tham Donald K., *CIMOSA: Enterprise Modeling*, Enterprise Integration Laboratory, University of Toronto

[Tham 1995] Tham Donald K. 1995 / PERA methodology . Enterprise integration laboratory, University of Toronto.

[Udell 1994] Udell J., *ComponentWare*, Byte Magazine, 19(5), pp 45-56, 1994

[Uschold 1998] Mike Uschold, Martin King, Stuart Moralee and Yannis Zorgios, *The Enterprise Ontology*, *The Knowledge Engineering Review*, Vol. 13, Special Issue on Putting Ontologies to Use

[Uschold 1996] Uschold Mike, Michael Gruninger, Ontologies: *Principles Methods Applications, The Knowledge Engineering Review*, Vol 11:2, 1996 93-116

[Vernadat 1994] Vernadat Francois B, *Business Process and Enterprise Activity Modelling: CIMOSA Contribution to a General Enterprise Reference Architecture and Methodology*, 3rd Annual Conference on Automation Robotics and Computer Vision, Singapore 1994

[Wang 1993] Wang W.,Popplewell K., Bell R., *An integrated multi-view system description approach to approximate factory modelling*. International Journal of Computer Integrated Manufacturing, 1993 6.3: 165-174

[Weston 1994] Weston R.H., Coutts I. A., *model enactment based on the use of the CIM-BIOSYS Integrating infrastructure*, Proceedings 3rd conference on Automation Robotics and Computer Vision (ICARVC'94), Singapore, 8-11 November 1994

[Weston 1994] Weston J. T., *Three dimensions of CIM*. Production and Inventory Management Journal, 1994, First Quarter: 59-61.

[Whitman 1997] Whitman L.E., Brian L. Huff, *Living Enterprise Model*. Automation & Robotics Research Institute, University of Texas. Industrial Engineering Research Conference Proceedings. 1997

[Whitman 1996] Whitman M. Michael, Michael L. Gibson, *Enterprise Modeling for Strategic Support*, Information Systems Management Journal, Spring 1996

[Whitman 1999] Whitman L.E., *A methodology for the classification of a living model of the enterprise*, Ph.D. Thesis, University of Texas at Arlington

[Wooldridge 1996] M. Wooldridge, M. Fisher, and C. Dixon. A Resolution-Based Proof Method for Temporal Logics of Knowledge and Belief. In D. M. Gabbay and H.-J. Ohlbach, editors,*Proceedings of the International Conference on Formal and Applied Practical Reasoning*. Springer-Verlag, June 1996.

[Wooldridge 2000] M. Wooldridge and P.Ciancarini. *Agent based software engineering: State of the art*, In P. Ciancarini and M. Wooldridge, editors, *Agent-Oriented Software Engineering*. Springer-Verlag Lecture Notes in AI Volume 1957, January 2000.

8.2 Bibliography

Anderson L. David, Frank E. Britt, Donovan J. Favre, The Seven Principles of Supply Chain Management,

Bernus P., Nemes L., Williams T.J., Architectures for enterprise integration, Chapman & Hall, London

Bouff L.A. John, *A Supply Chain Management Perspective*, Conference on Global Chain Management, Singapore, January 23, 1997

CIMCA'99 *Computational Intelligence, Volume 2, Information Retrieval* pp. 222-278 CIMCA'99, Conference Vienna, Mohamadian (ed)

Dimitris N. Chorafas (1996) Knowledge Engineering *van nostrand reinhold New York*

Clancey, W. J., "Heuristic classification", *Artificial Intelligence*, Vol. 20, pp. 28-37, 1985.

Cohen M.A., Lee H.L., *Strategic analysis of integrated production-distribution systems: models and methods,* journal of operations research, 36(2), pp216-228, 1988

Cohen M.A., Moon S., *Impact of production scale economies, manufacturing complexity and transportation costs on supply facility networks,* Journal of manufacturing and operations management, 3, pp269-292

Davis, B.R., Smith S., Davies M, *Integrated Computer Aided manufacturing (ICAM0 architecture Part III/Volume III: composite function model of Design Product,* Technical Report AFW AL-TR-82-4063 Volume III, material Laboratory, Air force Wright

aeronautical Laboratories, Air Force Systems Command, Wright Peterson Air Force Base, Ohio 45433, 1983

Dillon T, Tan P L (1993) *Object-oriented conceptual modelling* Prentice Hall,1993

Doumeingts G., Vallespir B., Chen D., *Methodologies for designing CIM systems: A surve,* Computers in Industry 25:263-280.

Edmonds A. Ernest, Linda Candy, Rachel Jones, Bassel Soufi, *Support for Collaborative Design: Agents and Emergence*, Communication for the ACM July 98'/ Vol 37, No.7

Evalds V, Tenteris J, Riga Information Technology Institute (RITI), *Proceedings of the Second International Baltic Workshop "Databases and Information Systems". Tallinn, June 12-14, 1996, Vol.2, pp.17-36*

Fadi G. Fadel, Mark S. Fox, Michael Gruninger, *A Generic Enterprise Resource Ontology Proceedings of the 3rd IEEE Workshop on Enabling Technologies;* Infrastructure for Collaborative Enterprises, April 1994

Fernadez Mariano, Asuncion Gomez-Perez, Natalia Juristo, *Methontology: From Ontological Art Towards Ontological Engineering,* Laboratorio de Inteligencia Artificial, Universidad Politecnica de Madrid, 1996

Fikes Richard, Mark Cutkosky, Jeffrey Van Baalen, *Knowledge Sharing Technology,* November 1991, Knowledge Sharing Technology, stanford University

Gomez-Perez Asuncion, *Towards a Framework to Verify Knowledge Sharing Technology*, Expert Systems with Applications, Vol. 11, No.4 pp519-529,1996

Guerino Nicola, *The Ontological Level*, 16th Symposium in knowledge engineering , Kirchberg, Austria 1993

Guiseppe Menga, Gaetano Messina, Guido Trocomi, *An Object Oriented Framework for Enterprise Modeling*, IEEE 1994

Hammond & Kristian, Case based planning: Viewing planning as a memory task. Boston Academic Press 1989.

Herbane Brahim, *Competence, Autonomy and Strategy in the British Automotive Components Industry*, Department of Corporate Strategy, De Montfort University, 1995

Herbsleb D. James, Rebbeca E. Grinter, Lawrence Votta, *Geographically Distributed Systems Development* Bell Labs, Lucent Technologies, University of Michigan

Hildum W. David, Norman M. Sadeh, Thomas J. Laliberty, John McA'Nulty, Stephen F. Smith, Dag Kjenstad, *Blackboard Agents for Mixed-Initiative management of Integrated Process-Planning/ Production Scheduling Solution Across the Supply Chain, Production Planning and Control Laboratory*, Center for Integrated Manufacturing Decision Systems & Software Engineering Laboratories

Hinkkanen Aimo, Ravi Kalakota, Porama Saengcharoenrat, Jan Stallaert, Andrew B. Winston, *Distributed Decision Support Systems for Real Time Supply Chain Management using Agent Technologies*, Enterprise Engineering, Remigijus Gustas, March 1996
Hough Deborah, *Modeling The Supply Chain: try the future now*, Conspectus June 1996

Karp Peter, *A Generic Knowledge Base Access Protocol*, Artificial Intelligence Center, SRI International May 1992

Kauffman M., Explorations in the representation of knowledge, Sowa John, San Mateo. 1991

Khoshafian S, Abnous R Object orientation: concepts, languages, databases, user interfaces Wiley, 1990

Kosanke Kurt, *Enterprise Integration and Enterprise Modelling*, CIMOSA Association Germany, International Conference on CAD/CAM Robotics and Factories of the Future 1995'

Kuo-Ming Chao, Marin Guenov, Bill Hills, Peter Smith, Ian Buxton, Chen Fang Tsai, *Sharing Domain Knowledge in Distributed Artificial Intelligence*, School of Computing and Information Systems, University of Sunderland, Engineering Design Center, University of Newcastle, Department of Industrial Management, Tamsui Oxford University College, Taiwan, 1996

Liebowitz J. (Ed), *Applied Expert System: The handbook of*, CRC Press1997Shelton R. Eugene, *An Object Oriented Method for Enterprise Modeling: OOEM*, Open Engineering, Inc. 1992

Mathe Natalie, James R. Chen, *User Centered Indexing for Adaptive Information Systems User Modeling and User Interaction*, Vol 6 pp225-261, 1996

Mildred L. G. Shaw, Brian R. Gaines, *Comparing Conceptual Structures: Consensus, Conflict, Correspondence and Contrast*, Knowledge Science Institute, University of Calgari,1989

Ming C. Hao, Deon Glajchen, Joseph S. Sventek, *SmallSync: A Methodology for Diagnosis & Visualisation of Distributed Processes on the WEB*, Hewlett Packard

Mujaba M. Shahid, *Enterprise Modeling and Simulation: Complex Dynamic Behavior of a Simple Model of Manufacturing*, December 1994 Hewlett Packard Journal
Sadeh E. Norman, Stephen F. Smith, Jay Swaminathan, *Supply Chain Management and Analysis*,

Perkins Alan, Enterprise Engineering, Visible Systems Corporation 1997 http://www.ies.aust.com/~visible/papers/EEMT.html

Peel R, Harmon K , Object Orientation and Business Driven Information Engineering, Visible Systems Corporation, 1997 http://www.ies.aust.com/~visible/papers/OO-IE.html

Petrie C. J., *Introduction In Proceedings of the First International Conference on Enterprise Modelling* Ed. Petrie, C. J., 1-17, MIT Press, Massachusetts, 1992

Pomberger G (1996) Object orientation and prototyping in software engineering *Prentice Hall, London*

Satzinger J W (1996) Object-oriented approach concepts modeling and system development *Boyd & Fraser Pub,New York*

Smith Peter, K-M Chao, T-F Chiu, C-F Tsai, *A Case Study of Using Technologies of Knowledge Sharing in Casting Production*, Lin SS, , School of Computing and Information Systems, University of Sunderland, Engineering Design Center, University of Newcastle, Department of Industrial Management, Tamsui Oxford University College, Taiwan, 1996

Smith Simon, *Towards an Enterprise Modeling Method for CSCW Systems*, Cornelia Boldyreff, Department of Computer Sciene, Durham University, IEEE 1995

Strick Linda, Jens Meinkohn, *Specifying Pan European Management Systems*, GMD-FOCUS, Handenbergplatz 1, D-10623 Berlin, F.R.G. 1994

Srinivasan K. Sundaresan J., *Comparison of Object Oriented Programming Languages: The Enterprise Modeling Framework Perspective*, Joop Journal June 1994

Swaminathan M. Jayashankar, Stephen F. Smith, Norman M. Sadeh, *Modeling Supply chain Dynamics: A Multiagent Approach*, Haas School of Business, University of California, Berkeley, April 1996

Swaminathan M. Jayashankar, Stephen F. Smith, Norman M. Sadeh, *Sharing Supplier Available Capacity Information,* Haas School of Business, University of California, Berkeley, January 1997

Sulin Ba, Andrew B. Whinston, *An Enterprise Modeling Approach to Organizational Decision Support, Center of Information Systems Management*, IEEE 1995

Williams T.J., *The PURDUE Enterprise Reference Architecture*, Purdue laboratory for applied industrial control, Purdue University. 12th Triennial World Congress, Sydney Australia

Wither Eckart, Henrik Eriksson, Mark A. Musen, *Plug and Play: Construction of Task Specific Expert-System Shells Using Sharable Context Ontologies*, Medical Computer Science Group, Knowledge Systems Laboratory, Stanford University, April 1992