University of Newcastle Upon Tyne

Department of Computing Science

Enterprise Modelling And Its Application to Organisational Requirements, Capture and Definition

by

Andrew J. C. Blyth

NEWCASTLE UNIVERSITY LIBRARY

095 50464 4

Thesis LP87

Ph.D Thesis in Computer Science.

Winter 1995.

Abstract

Computers have gone from being solely large number crunching machines to small devices capable of performing a myriad of functions in a very small space of time. Computers are now used to control just about every facet of daily life; they can now be found in automobiles, washing machines and home heating systems. This rapid diversification brings a great many problems. Traditional software engineering methodologies are failing to meet and address these new problems. The goal of this thesis is to develop a new approach to organisational requirements engineering. A new modelling approach to representing organisations will be developed which will draw upon the concepts of a systems architecture, modelling the life cycle of responsibilities and the execution of conversations. Using this architecture an organisation will be able to embed social and cultural aspects within the modelling notation. From the modelling of responsibilities a clearer picture of the organisation's aims, objectives and policies will be developed along with a definition of what objects and access rights are required in order for the organisation to function. Using speech act and Petri net based models to model conversations a clearer understanding of the dynamics and constraints governing organisational behaviour can be developed. These modelling approaches are then applied to two real life case studies in order to demonstrate and evaluate their performance and usefulness.

Chapter 1 *The Problem*

1.1 The Problem

"At this moment in time two sets of values present in society are in conflict with each other. On the one hand we have a powerful technical value system which tells us to make maximum possible use of technology so that we may become wealthy and comfortable. On the other hand we have a humanistic value system which tells us to beware of technology for it is a mirage which will lead us to disaster rather than success. Somewhere in between these two value systems is another which says technology is essentially neutral; whether it produces gains or losses depends entirely on the decisions that are taken on how it shall be used" (Mumford & Hedberg, 1975).

"Two things have influenced organisations more than anything else; the coffee machine and the computer. The reason for this is that they both mediate social relationships" (Ehn, 1989).

"The problem facing system designers today is to balance these two value systems. In the early 1980's General Motors embarked upon an enormous investment in automation. In 1985, it opened its new showcase factory in Detroit, Michigan. This new factory had 50 automatic guided vehicles to ferry parts around the plant, and 260 robots to weld and paint. Almost a year after the plant was open it was still only producing about half the number of cars that it was supposed to. The company has rushed to adopt a high technology solution to a problem that is social in nature" (Wiener, 1993).

From the above three commentaries we can observe that requirements in general come from one of two domains. The first domain called the problem space domain is concerned with organisational policies and goals (the social value system). From these policies and goals a set of requirements that will influence the specification can be derived. The second domain called the solution space domain is

concerned with design and implementation details and decisions (the technical value system). From having to consider the design and implementation of the system further requirements can be derived.

Problem domain requirements are in general intangible social concepts (Dahlbom & Mathiassen, 1994) that are mapped into a solution space by means of the system specification. This mapping can often lead to, and cause, ambiguity as was shown in (Gause & Weinberg, 1989). Through the specification the system exhibits some behaviour that has to be validated. This validation is an interpretation back into the social domain of the problem. Consequently errors can occur at the various stages in the process of capturing, expressing, validating and interpreting the requirements. This process however is still viewed by most software engineers as a mechanical one rather than a social and linguistic one.

1.2 The Nature of Specifications

1.2.1 Introduction

The core material stated here is taken from (Dobson & McDermid, 1990). It has been stated that an appeal to a dictionary is the last resort of a desperate man. However as the word "specification" has been used in an engineering context for many years, it seems relevant to consider the dictionary definitions of "specification". The following two definitions are taken from the Oxford English Dictionary.

> *Specific definition or description.*
>
> *A detailed description of the particulars of some projected work in building, engineering, or the like, giving dimensions, materials, quantities, etc., of the work together with directions to be followed by the builder or constructor.*

The first definition uses terms like *specific* to imply that a specification should be a precise definition of the system. The second definition uses terms like *particulars* to imply that general properties for some system objects can be assumed and thus need not be defined. The term "*directions to be followed by the builder*" is part of the specification of the development process rather than the product. Hence a specification may consist of requirements that will be placed not only on the product but also on the process.

1.2.2 Views of Reality

A specification can be viewed as a medium through which people communicate about the problem and its possible solutions. This communication is achieved through the use of one, or many different, languages. As a result we are now forced to ask the question "How do we select what language to use in order for us to communicate through our specification?" The problem that we have with languages such as English is that they are informal, inconsistent and imprecise. The problem that we have with formal languages such as Z (Spivey, 1989) is that they are too formal, too restrictive in expressiveness.

In (Wittgenstein, 1958) it is argued that "the meaning of an expression is in its usage". When we apply this idea to the problem of constructing and validating a specification we see that in order for all parties to communicate through the specification we need to build up a joint understanding of what an expression means. This process of building up an understanding and ability to use a language is termed a language game in linguistics. This process involves its users in (i) learning how to map their experiences and values (their view of reality) into the language and (ii) validating that mapping through exercising the language. In applying this view of the use of language we can still view the task of constructing and validating a specification as one of constructing and validating a socially agreed view of reality.

The question we are forced to ask when viewing reality is "how do we view, interpret and manipulate the objects that exist within our reality?" We can observe that three types of interpretation seem to exist. These are (i) Syntactic in how we view an object, (ii) Semantic in how we interpret that view and, (iii) Pragmatic in how we use that view. The interpretation that an individual places on a view of an object is a socially defined one, and one that is unique to the individual.

1.2.3 Specification as a Communication Medium

As has been stated often by experienced software engineers, a specification document is used as a means of communication between a group of people. For simple systems a specification may serve as a communication medium between the customer (problem owner) and the supplier (problem solver). However for more complex systems such as an SDI system a specification document will serve as a communications medium for a whole host of people such as users, customers, designers, evaluators and analysts. As was shown in (Zachman, 1987) each user of the specification will interpret it in a different way, since for each user the specification will serve to define a different set of objectives and goals. As the number of people

who use the specification as a communication medium grows so it will become more difficult to ensure that all the users of the specification have compatible interpretations of the specification.

As was stated in (Dobson & McDermid, 1990) each user of the specification will make an interpretation function into their own domain of knowledge. For example a problem owner will make an interpretation into the problem domain. In contrast a designer solver will make an interpretation into the solution domain. The task of the problem solver can be seen as providing a bridge between these two domains. The problem is made even worse when one considers the nature of an interpretation function. An interpretation function may be said to consist of three sub interpretation functions.

- The Syntactic Interpretation Function.
- The Pragmatic Interpretation Function.
- The Semantic Interpretation Function.

The syntactic interpretation function takes place within a formal, well defined framework. Typically, within an engineering context this framework takes the form of a logical formalism. This function may be refined and examined by means of logical argument and thus issues of consistency and omission discussed. The syntactic interpretation function is semiotical in nature and as a result is independent of the sociological and cultural domain from which the specification of the system is derived.

The pragmatic interpretation function is concerned with the purpose, effects and implications of the object. The semiotical component is concerned with the effects and implications of the object and is derived from the heuristics of the person making the interpretation function. The hermeneutical component is concerned with the purpose of the function on the object. Consequently the pragmatic interpretation function is both semiotical and hermeneutical in nature.

The semantic interpretation function is, however, a function that is much harder to understand, as this function is based upon a mutual understanding of a set of common concepts, the concepts being either logical or social in nature. If the concepts are social in nature then the environment within which the role holder and problem are placed, and the influence that the environment exerts on the role holder as well as the problem itself, will come into play. In essence the semantic interpretation function

is one of negotiation to achieve a common understanding of a set of concepts. This function differs from the syntactic interpretation by being hermeneutical in nature.

The questions that we are now forced to face is " how do we bring these three interpretation functions together within a framework and ensure that they function as a medium of communication?" An architecture functions as a medium of communication and supports all three interpretation functions (Zachman, 1987). An architect uses an architecture to map from the problem domain to the solution domain by an architect (See Figure 1.2.3.1). The architecture functions as a medium of communication through which a) the architect and the problem owners can explore and define the problem space, and b) the architect and the problem solvers can explore and the define the solution space.

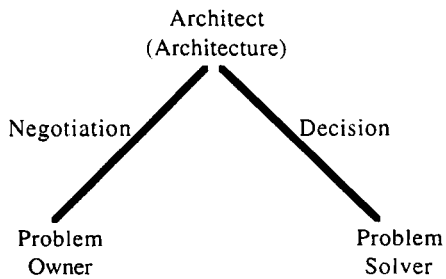


Figure 1.2.3.1 Uses of an Architecture

It is important to note that the architecture itself lies in neither domain and thus is neutral to both the problem and the solution. The mapping from the problem to an architectural view of the problem is one of negotiation as a common interpretation function of the architectural view of the problem has to be achieved. The mapping from architecture to solution is one of decision. The decision is to decide how a given set of architectural views of the problem should be implemented. At the heart of both of the negotiation and the decision processes the architecture is functioning as a medium of communication (Zachman, 1987).

To understand the problem of interpretation and negotiation we will define a specification to be a set of architectural views of the system, where a view is defined as an instantiation of a person's, or organisation's, perspective or viewpoint on the problem or its solution. Rather than just have a random set of views on the system we need to be able to classify and strcuture them. In addition, we also need to provide an architecture through which we can an instantiate, reason about, and unify the views within a particular classification. Rather than have a random set of classifications we need a structured set of classifications in order to direct our thoughts and our activities. In (ANSA, 1990)a structured set of five classifications are presented and outlined. Each classification within (ANSA, 1990) is referred to as a projection and

defines a particular set of issues and concerns. If we view a system as a complex object that exists within the real world, then a projection can be viewed as a torch that we shine on the object in order to elicit and instantiate a certain set of requirements. For every projection there are many possible architectures that we may use. In the ANSA Reference Manual five projections of a system are presented along with their particular associated architectures.

- The Enterprise Projection
- The Information Projection
- The Computation Projection
- The Engineering Projection
- The Technology Projection

The principal role of the enterprise projection is to interpret the role of a computer system within an organisation. The enterprise projection describes the overall objectives of a system in terms of agents, actions, resources and policies. The enterprise projection facilitates the problem solvers in the understanding of the organisational context within which activities are performed. The enterprise projection provides a clear description of enterprise within which the system is to be placed. It makes explicit a) the roles and interactions that can exist within the organisational system, and b) the flow and mediation of social relationships that the system is required to support. Thus systems designers can examine and explore the implications of the system design at an organisational level. Design decisions made using the enterprise projection are concerned with the role of the system. The requirements that are expressed in the enterprise projection are expressed in both system and environmental terms.

The primary function of the information projection is the identification and location of information, and the description of the information processing activities. The information projection facilitates a system designer to express the structure, interpretation and timeliness as well as to examine the relationships that can exist between information objects within the organisation. It will also facilitate a system designer to express a) how the information is manipulated and represented within the organisation, and b) how the information flows through and is mediated by the organisation. The projection also facilitates the expression and unification of multiple views of the information resources and information processing activities. System designers can use the information projection to explore and examine the nature and

role of information within the organisation as well as how the information flow within the organisation can affect that organisation. The requirements that are expressed in the information projection are concerned with the items of information and information processing activities.

The purpose of the computational projection is to provide a framework for the description of the structure, specification and execution of application programs. An application program is a specification of some activity that is suitable for execution on a computer system. The computational projection concentrates on the problems presented by the execution of applications on computer systems. The description of the computational projection is in terms of the information representations, programming languages, systems services and program specification. Thus the projection details how the applications should be structured, modularised and parallelised.

The primary function of the engineering projection is to provide a framework for the specification of the mechanisms and tradeoffs required to support the application programs. Thus the purpose of the projection is to describe the organisation, and the system within it, in such a way that the system designers can reason about its performance. The projection provides a set of system building blocks and explains how to interconnect and organise them in an efficient way, so that the relevant application requirements can be met. A designer can use the engineering projection to examine and explore the trade offs between such issues as performance, dependability, security, maintainability and scaling. System designers, managers and the maintenance staff will all use this projection as a basis upon which to forecast the consequences of possible system configurations.

The primary function of the technology projection is to express the system in terms of its physical components, hardware, software etc. Thus in the technology projection issues concerned with construction and maintenance can be expressed and discussed.

1.3 The Thesis and its Validation

1.3.1 The Thesis

The hypothesis presented is that there is a set of organisational requirements on an information system which is usefully identified and modelled by means of enterprise and information modelling architectures. In particular I assert the following:

- That it is possible to create both an enterprise and an information modelling architecture, using social and linguistic constructs such that it is possible to map concepts between the two architectures.

- That it is possible using these architectures to construct and validate the structural and dynamic enterprise and information models through a communicative dialectic process involving both the problem owners and the problem solvers.

- The result of this mapping and modelling is that greater clarity and precision of the organisational context of the problem and its possible solutions can be developed.

1.3.2 The Validation

1.3.2.1 Introduction

The validation of this thesis will take the form of two case studies. Both case studies are drawn from problems which real organisations are experiencing today. In addition, both case studies are drawn from different areas of industry so as to demonstrate domain diversity. The approach taken to ascertain whether greater clarity or precision can be achieved will be different for each case study, and each approach and case study is outlined below.

1.3.2.2 The Man in the Van Case Study

The first case study is taken from a regional electricity board. The electricity board is responsible for supplying electricity and electrical services to a large region, covering several large suburban and rural areas. The validation of the hypothesis against this case study will take the following two forms:

- A structural and dynamic enterprise model of the current system will be developed using the ideas and notations presented in this thesis. This model will then be compared and contrasted with a model of the current system that was developed using various business analysis techniques (Eason, 1995; Eason, 1985; Eason, 1987).

- A structural and dynamic enterprise model of the system as proposed will then be developed and this is compared with the enterprise model of the current system in an attempt to ascertain whether the models contain any unexpected

and significant results or differing views, and also to see if any information was lost.

1.3.2.3 The Accident and Emergency Department Case Study

The second case study is taken from an accident and emergency (A&E) department of an inner city hospital and was undertaken as part of the ORDIT project. This hospital is one of several that contains an A&E department in the wider metropolitan area and thus serves as a front line institution for the delivery of primary and secondary health care. The validation of the hypothesis against this case study employs the following three components:

- An information model of the current system will be developed and derived using the ideas and notations presented in this thesis. This model is then compared and contrasted with a model of the current system that was developed using various business analysis techniques (IT-Uptake Project, 1986a; IT-Uptake Project, 1986b; Lorenzi,Riley,Ball, & Douglas, 1995) to see if the information model could: a) capture all the data contained in the various business analyses, and b) show up any contradictions and anomalies.

- A dynamic model of the current system will be developed and examined to see if the dynamic model can: a) capture all the data contained in the various business analyses, and b) show up any contradictions and anomalies.

- Both sets of models will then be presented back to the analysts and consultants that originally worked on the problem and their comments and suggestions elicited.

1.4 The Solution and its Claim of Originality

Chapter one begins with a general problem definition statement of the problems concerned with requirements engineering. In this section the concept of problems as being social in disposition is introduced (Ehn, 1989; Ross, 1977). In the following section it is argued that the requirements engineering process is social in nature: when one is constructing a specification, one is constructing a social view of reality. The nature and purpose of a specification is explored in the next section. The concept of a specification as a communication medium (Zachman, 1987) is defined and explored. In the final sections I state what my hypothesis is and how I propose to prove it. A statement of the content of the thesis is given along with a summary of how and where the work contained in this thesis draws upon other work.

We start Chapter two with a set of classification and evaluation criteria with which the requirement engineering notations, techniques and methodologies will be graded. In the following section an overview of various methodologies such as SSM (Checkland, 1986), SADT (Ross & Schoman, 1977), SSADM (Nicholls, 1987) and CORE (Mullery, 1979) are presented. In addition, this section finishes with a general discussion on the methodologies currently available to an information technology analyst/engineer. The next section contains an overview of various requirements specification language and notations such as PSL/PSA (Teichroew & Hershey, 1977), HOS (Hamilton & Zeldin, 1976), RSL (Alford, 1977) and DFD (Dennis, 1974). This section terminates with a review of several other notations and languages not covered earlier in the chapter and a discussion of the various merits of those various languages and notations.

Chapter three begins with an investigation into, and a classification of, the philosophical assumptions behind any modelling approach. In the following section the concepts of what an architecture is, and how it might be applied to the software engineering process is elucidated and developed (Perry & Wolf, 1992; Zachman, 1987). The structuring of an architecture to aid the process of requirements engineering is discussed and outlined in the next section. In the following section the concepts of enterprise and information modelling architecture are discussed and elucidated. The enterprise conceptual framework was developed as part of the ORDIT[1] project and published in (Blyth,Chudge,Dobson, & Strens, 1993; Blyth,Chudge,Dobson, & Strens, 1993). In the next section a solution to the problem of mapping between these two modelling frameworks is set forth. This solution is then used within a small example in the next section. Apart from the enterprise conceptual framework all of the work contained in this chapter is my own.

We start Chapter four with an examination of the organisational issues that current methods have failed to address adequately to some degree. The meaning and purpose of a method is then examined, along with the methodological, technical and sociological context within which the method defined in this thesis is required to function. In particular, attention is paid to the various components that make up the process model and the various projections that can be used within the process model to describe and analyse the system. This process model was developed as part of the ORDIT Methodology and is taken from (Blyth,Chudge,Dobson, & Strens, 1992). These projections are taken from (ANSA, 1990).

[1] The ORDIT (Organisational Requirements Definition for Information Technology) project is an ESPIRT II project, Number 2301.

Chapter five begins with a discussion of what an organisational model is and how we might use it. The next sections describe in depth the various components and concepts that make up an enterprise projection diagram. This work is my own but drawn upon (Dobson & McDermid, 1988; Dobson & McDermid, 1990; Dobson & McDermid, 1991) and has been published in (Blyth et al., 1993; Blyth et al., 1992; Blyth et al., 1993) The final sections describe in depth the numerous components and concepts that make up an information projection diagram. The work contained in Chapter 5 is work that I have performed on my thesis by myself. I would however like to acknowledge the help and input from my colleagues on the ORDIT project.

Chapter six begins with a brief discussion and overview of the various notations that are currently available to model and analyse organisational dynamics. A graphical model of conversations which draws upon the concepts of speech acts (Austin, 1962; Searle, 1969) and directed graph theory is then presented. This graphical model has been published in (Blyth & Chudge, 1993; Blyth & Chudge, 1994). I decided to use coloured Petri nets as the executable formalism for executing the conversation models. As a result of this decision I present an in depth overview of coloured Petri nets (Jensen, 1986; Murata, 1989). A mapping from each component in the graphical conversation model to a coloured net construct is then defined along with a set of projection token types. Together both modelling techniques form a framework from within which organisational dynamics can be defined and reasoned about. The work contained in Chapter 6 is work that I have performed on my thesis by myself.

Chapter seven begins with a problem statement for the first case study (The Man in the Van). A set of enterprise diagrams is then constructed and analysed for the current system. A set of conversation diagrams is then derived from the perspective of a day electrician. The conversation diagrams are constructed so that the process of job allocation and job completion can be analysed. From the conversation diagrams a picture of the flow and life cycle of the organisation's responsibilities is constructed and analysed. In addition, the objects and access rights over those objects needed by the electrician is identified. These models have been published in (Blyth & Chudge, 1993; Blyth et al., 1993; Blyth et al., 1992). All of these models for the current system are then compared and contrasted with the models of the current system that were developed using various business analysis techniques. A set of enterprise and conversation models is then constructed for the system as proposed. Then the models of the current system are compared with the models of the proposed system in an attempt to ascertain whether the models contain any unexpected and significant results or differing views.

Chapter eight begins the second case study (The Jervis Street Hospital) with a problem statement. After this a small enterprise diagram of the organisation is constructed. This model has been published in (Blyth & Chudge, 1995). From this enterprise diagram a contractual view of the system is derived and validated. The contractual model of the system is then compared and contrasted with a model of the current system that was developed using various business analysis techniques in an attempt to validate the modelling techniques. A set of conversation diagrams is then constructed for the junior nurse in the role of administerer of drugs. From these conversation diagrams a coloured Petri net model is derived and a set of formal requirements elucidated. These diagrams are then analysed to see if they contain any contradictions and anomalies. Finally the models created in the second case study are presented back to the original consultants who worked on the problem.

Chapter 9 begins with a restatement of the hypothesis and its method of validation. This is followed by a summary of the work contained in the thesis. In the next section the issues of suitability and applicability to the problem are discussed. In this section issues such as domain suitability and traceability are examined. Chapter 8 finishes with section 8.3 in which some further work is identified.

In appendix A a set of operators which range over enterprise and information diagrams is defined in Z (Spivey, 1989). The purpose of these diagrams is to facilitate in the formal expression and reasoning about enterprise and information diagrams. In appendix B a set of rules governing the meaningfulness of the enterprise and information diagrams is defined. The objective of the rules is to allow for the formalisation of a set of axioms which can then be applied to an enterprise and information diagram. These axioms can be defined or modified by either the problem owners or problem solvers and this process forms an integral part of defining the meaningfulness of a diagram. In appendix C a set of operators is defined through which the meaningfulness of the notation used to model organisational dynamics can be reasoned about. At the end of this appendix the operators are applied to a requirements statement to show how it can be formalised and validated with reference to a model.

Chapter 2 *Survey of Organisational Requirement Engineering Techniques*

2.1 Introduction

Information technology systems are currently failing because of the lack of understanding of social concepts that an information technology system is required to support. This lack of understanding contributed to the failure of the following systems: the London Ambulance Service (South West Thames Regional Health Authority, 1993), General Motors car manufacturing plant in Detroit, Michigan (Wiener, 1993) Jacksonville General Hospital, Florida (Fox, 1995).

The weakness of most requirement capture and analysis techniques is that they fail to help the problem solvers understand the nature of the problem correctly by not helping them to understand the way in which information technology systems interact with their environment and thus the way in which they change and evolve with their environment (Eason, 1988). In (Stamper, 1985) the point is made that garbage in leads to garbage out and consequently it is vitall that when starting the requirements engineering process we start from the correct point and capture and feed-back valid and appropriate requirements.

2.2 Framework for Examination of Requirements Engineering Processes

2.2.1 The Purpose of a Classification

The purpose of classifications is to allow us to identify the strengths and weaknesses of current techniques that are being applied to the oganisational requirements for information systems. In a more global sense the purpose of a classification is to allow us to say something about the object that we are examining.

The choice of a classification schema therefore needs to be pertinent to the area of examination and needs to distinguish between other objects.

"*Organisations are dynamic entities - continually having to change and evolve*" (Blyth,Chudge,Dobson, & Strens, 1993).

"*Two things have influenced organisations more than else, the coffee machine and the computer; why, because they mediate social relationships*" (Ehn, 1989).

As a result of the reason stated above I have chosen the following two classification schemes. The ANSA framework for systems development (See Section 2.2.2) presents five autonomous views of an information system. Through these views we can build up a picture of how an organisational information system might be realised in an information technology system, and what impact that information system might have on the organisation. The Axiological framework (See Section 2.2.3) allows us to classify, examine and explore the social value system and its interactions, which an information system will be required to support. The reason for choosing these two classification schemes is that I believe that current requirements engineering techniques for organisational information systems have failed to address the two areas that these frameworks address.

2.2.2 ANSA

The work presented in (ANSA, 1990) defined five different ways of viewing a system (See Figure 2.2.2.1). Each of these different ways of viewing a system can be seen as a projection onto a particular set of concerns. The five projections that were identified were the enterprise, information, computation, engineering and technology projections, where a projection can be thought of as a particular way of viewing the problem, or perspective on the problem. The description of the problem in each projection is self-contained and the difference between projections is not how much of the system they describe, but rather to what they give emphasis, as each projection reveals different facets of the system.

ANSA	Enterprise	Information	Computation	Engineering	Technology

Figure 2.2.2.1 The ANSA Projections

Each projection represents a particular viewpoint, with its own set of issues and concerns. It is important to realise that all the projections of the system should be examined, and it is a mistake to believe that some projections are more fundamental than others. Each projection represents a particular viewpoint by defining an arena within

which a set of questions and their solutions can be examined and explored. The projections allow a problem solver to explore various aspects of the problem, from the abstract issues of responsibility, information flow, manipulation and transformations to what kind of architectural platform the proposed system could use. These projections thus provide the problem solver with a powerful tool to aid in the task of requirements engineering. So by examining how various methods, techniques and notations allow us to express the concepts described in the five projections we should get a better understanding of how each methodology, technique and notation fits into the process of requirements engineering.

A brief outline of the five ANSA projections was given in Chapter 1 Section 1.2.3. A more detailed description of the projections, and the purposes that they serve within the requirements engineering process is presented in Chapter 3. The definitive definitions of the projections are however given in (ANSA, 1990).

2.2.3 Axiological

In (Dobson, 1988) the word *axiology* is defined to mean the theory of values and, in particular, a study of the ways in which values provide a basis for judging and deciding on actions. The purpose of an axiological framework is to facilitate the problem solvers and problem owners in their exploration and comprehension of the problem and the solution. The axiological framework can be used to express and explore the nature of the values, along with the values themselves, that the solution is required to encompass and encapsulate. Through the use of this axiological framework we can examine social concepts such as responsibilities, obligations and commitments, and examine how, when and where they are manipulated.

Axiological	Conceptual	Logical	Log. Ops	Log. Const

Figure 2.2.3.1 The Axiological Framework

This framework is divided into four sections, where each section has its own concepts, goals and axioms for reasoning and expression. The first section bestows a structure within which it is possible to cultivate an understanding of the nature and structure of the concepts to be embedded in the system. The second section provides a logical framework within which it is possible to explore the values that the system is required to encapsulate, along with a logic to reason about the values. The third section delivers a framework from within which it is possible to understand and explore the logical operators that are required to manipulate the values. The fourth and final section

bestows a framework within which it is possible to examine a) the logical constraints under which the values encapsulated in the system will be required to function, and b) the effects of the logical constraints on the system.

2.2.4 Structure of Review

Over the years many requirements engineering methods, tools and techniques have been developed to address the problem of organisational requirements capture and definition for information technology. The rest of this review is divided into two sections. In the first section (Section 2.3) I shall review only certain organisational requirements engineering methods. These methods are chosen for the fact that a) they are in serious use today, or b) they represent a certain domain of requirements engineering methods. Similarly in the second section (Section 2.4) I shall review only certain organisational requirements engineering notations. These notations are chosen for the fact that a) they are in serious use today, or b) they represent a certain domain of organisational requirement modelling notations.

2.3 Methods for Requirements Engineering

2.3.1 The Soft Systems Method

The Soft Systems Method (SSM) approach to design was initially developed by Checkland (Checkland, 1986). The soft systems method can be seen as a general problem solving approach appropriate to human activity systems. The essence of the approach is an appreciation that for most problems, there are a number of problem statements that may be appropriate, and that the appropriateness of different solutions is largely determined by the particular viewpoints of those people who have an interest in the problem. The soft systems approach provides a conceptual framework to aid in the understanding of how the proposed system will function, along with how the proposed system will be defined. For example, a social security benefits system could be viewed as a system for ensuring that the needy obtain the benefit they deserve, or alternatively it might be viewed as a system for the protection of the "public purse".

Soft system thinking differs most from other approaches based on so called "hard" systems thinking (See Section 2.3.10), in that it allows the problem solver to explore the fuzzy and ambiguous nature of a problem and provides a means to discuss change. Soft systems thinking can be seen as the general case for which hard systems thinking is a special case. In soft systems thinking conceptualisation becomes system design, if the problem is sufficiently well defined. Improving the conceptual model

sharpens up into optimisation of a quantitative model, and implementing some variety of change becomes implementing a designed system.

The viewpoint that one adopts can strongly influence the kind of system that would be thought appropriate, and where such multiple viewpoints coexist in an organisation, a good design will be one that finds a suitable compromise between alternative views. A key feature of Checkland's methodology is the representation of such complex design environments in a suitably '*rich*' way, i.e. rich pictures, so that conflicts of interest can be identified and resolved, or a compromise problem statement reached.

In addition, this method demands that a system oriented approach to design be taken, where design is viewed as the creation of a formal system which must have certain features in common with other systems. These features include having a purpose or mission, a measure of performance, a decision taking process, components as subsystems, a degree of connectivity, an environment, a boundary, resources, and some degree of continuity. Emphasis is placed on describing possible systems in logical terms i.e. with regards to the 'what', rather than the 'how' of what should be achieved.

Checkland's approach has led to considerable debate within systems analysis circles, and appears to be useful where there is a complex design problem. Whilst it is difficult to obtain a measure of its acceptance, it has been utilised in a number of major projects, although it has not replaced hard systems design methods in any standard methodology. It is however beginning to be more widely used within the U.K. software industry.

Checkland's soft systems methods in general allows us only partially to identify, represent and validate issues associated with the enterprise and information component of the ANSA framework (Section 2.2.2). This is achieved through the construction and analysis of a set of rich pictures. This approach does not allow us to identify the computation, engineering or technology components of the ANSA framework. These rich pictures allow us partially to identify, represent and validate all the components from the axiological framework (Section 2.2.3). SSM does not direct its users in a particular direction when modelling, rather it is left up to the discretion of the user of the method.

2.3.2 SADT

The Structured Analysis Design Technique (Ross & Schoman, 1977; Tse & Pong, 1991) (SADT) was designed to aid a problem solver in the task of requirement elicitation and analysis. The method is designed to terminate with the production of a

specification that can then feed as the starting point into other standard methods. The basic concept behind SADT is that of slow refinement using a set of tools expressly designed to facilitate the process of requirements engineering. The tools vary from process models designed to provide a framework from within which it is possible to capture, define and examine requirements to mental models from within which it is possible to discuss conceptual issues. The SADT methodology is divided up into three stages as shown in Figure 2.3.2.1.

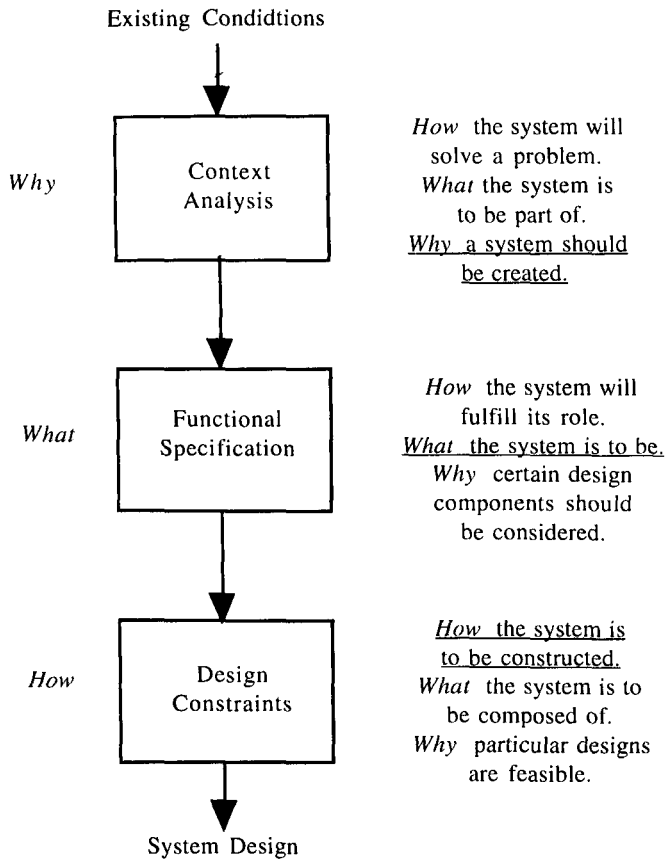


Figure 2.3.2.1 The SADT Stages

The context analysis stage serves to define the reasons why the system is to be created along with why certain technical, operational and economic feasibilities are the criteria which form the boundary conditions for the system. The functional specification stage acts as the description of what the system is to be, in terms of functions that the system must accomplish. Since this is part of the problem statement the functions must only be presented in terms of their pre and post conditions. The design constraint stage serves to summarize the conditions specifying how the required system is to be constructed and implemented. Thus the SADT methodology starts at the requirement elicitation phase of the development cycle and finishes by producing a specification that could then feed into a system design phase.

An SADT specification is made up of a hierarchy of diagrams. At each level of diagramming a network of nodes and arcs is formed, see Figure 2.3.3.2. The arcs connecting the nodes together are used to represent input, output, control and mechanism. A node is used to denote an activity of the system. In SADT a hierarchical specification can be built up in a top down fashion according to strict syntactical and semantic rules. Each node can also have a piece of text associated with it; this textual language can be natural or artificial in nature. SADT provides a nice graphical framework from within which problems can be refined and solutions expressed. SADT suffers from the graphical notation that it uses being too rich and thus too difficult to understand at a single glance.

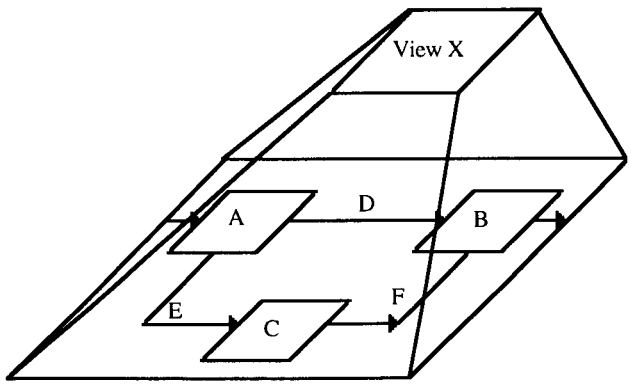


Figure 2.3.3.2 An SADT Specification.

While it is true that SADT was designed more for hard systems thinking than soft systems thinking, it does draw upon some of the ideas present in soft systems thinking. SADT makes use of the idea of a view of the system, as a way of examining and solving contradictions that may lie within the proposed system. SADT also makes use of a graphical language to express and validate sections of the proposed system.

EDDA (Trattnig & Kerner, 1980) is an attempt to improve SADT by adding a mathematical formalism, thus a specification could have its static and behavioural properties analysed. EDDA exists in two forms, a graphical from called G-EDDA and a textual form called S-EDDA. The formalism upon which EDDA is mounted is Petri nets, and extended Petri nets are a graphical from of EDDA, (G-EDDA). For every SADT specification it is possible to produce a corresponding G-EDDA and S-EDDA specification.

RML, requirements modelling language (Greenspan, 1984) is a formalism based on set theory and first order predicate logic that has also attempted to improve SADT by providing a language more akin to Pascal and C which allows a system designer to express requirements. By providing a means by which it is possible to

express requirements using such a language, the problem solvers are straying from the domain of abstract specification into the domain of abstract implementation, and thus the problem solvers may find themselves specifying the implementation of the system rather than the system itself.

SADT will only let you capture, represent and validate issues contained within the information and computation components of the ANSA framework (Section 2.2.2). The reason for this is that SADT views the world as a set of communicating processes. SADT does not directly address any of the issues contained in the axiological framework (Section 2.2.3). The identification, representation and validation of requirements from the axiological framework is left up to the user of the method.

2.3.3 SSADM

The SSADM (Structured Systems Analysis and Design Method) method (Ashworth, 1991; Ince & Andrews, 1991; Longworth, 1989; Nicholls, 1987) is a structured methodology in that it attempts to address four questions that continually arise in the process of systems development.

- What is the system to do ?
- When should it be done ?
- How should it be done ?
- Where is the information to be recorded ?

SSADM, whilst complementing other development activities, does not encompass them all. Each stage is broken down into a small set of steps which define the inputs, outputs and tasks to be performed. The product of each step and the interface between each step is clearly defined in the SSADM documentation. Each step serves to define a clear set of methods that can be used to achieve the overall goal of that step. The stages in SSADM follow a clearly prescribed linear sequence and SSADM methodology forces a problem solver to follow this sequence. SSADM starts at stage one *Investigate Current System* and finishes in stage six *Physical Design*. Figure 2.3.3.1 entitled *The SSADM Method*, describes the six stages that a problem solver would engage in using the methodology.What is not shown in Figure 2.3.3.1 is that from any stage there is a set of feedback loops back to all of the stages that have gone before.

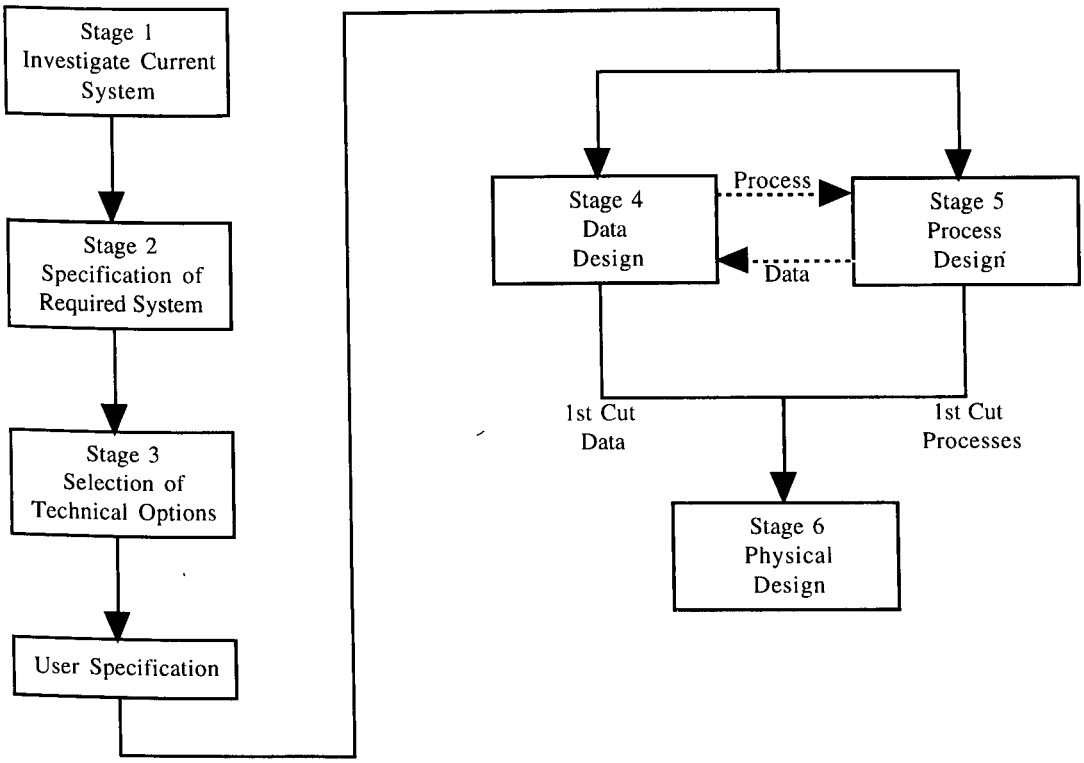


Figure 2.3.3.1 The SSADM Method

The first stage in the SSADM method is called *Investigate Current System*. This serves to allow the problem solver to learn the terminology and function of the system user environment. It will allow the problem solver to see how the current system works along with examining the data flows that exist within the system, and the informational concepts which the new system will need to support. The second stage is called *Specification of Required System*. In this stage the problem solver will construct a specification of the new system. This will be based on the information that was gathered in stage one and in consultation with the problem owners. The third stage, *Selection of Technical Options*, serves to generate a set of possible technical solutions to the problem. Each solution will be carefully costed out and evaluated against each other. Stage four, *Data Design*, will define the logical data design of the system such that all the required data will be included in the system. Stage five, *Process Design*, develops the definition of the system defined in stage two to a low level of detail so that an implementor can be given the necessary details to build the system. Finally in stage six, *Physical Design*, the complete data and process designs are converted into a design that will run on the target machine.

SSADM also employs a set of techniques (Ashworth, 1991; Ince & Andrews, 1991) that can be used at various stages in the methodology life cycle model. The rules of the syntax and notation of each technique are supplemented with guidelines on how it

should be applied in a particular step. The diagramming techniques that can be used within SSADM are Data Flow Diagrams (DeMarco, 1979), Logical Data Structuring, Entity Life Histories and Logical Dialogue Design. In addition SSADM also has a set of non-diagrammatical techniques including Relational Data Analysis, Quality Assurance Reviewing and Project Estimating. The SSADM methodology as it stands is a soft system methodology, in that it recognises that for any problem there are a number of different design solutions that may be appropriate, and that the appropriateness of different solutions is largely determined by the particular viewpoints of those people who have an interest in the problem and its solution. The methodology demands that a system oriented approach to design be taken, where design is viewed as the creation of a formal system which must have certain features in common with all other systems. Obviously a key feature in this methodology is the ability to represent and detect conflicts of interest from the viewpoint holders.

The SSADM methodology allows a problem solver to examine and explore the information and computation projections defined in the ANSA framework (Section 2.2.2), as it views an organisation as a set of conduits though which information flows and is transformed. It does not provide any framework for the discussion or analysis of any of the axiological concepts, operators or constraints (Section 2.2.3) that a proposed system would be required to encapsulate.

2.3.4 CORE

CORE is a method for COntrolled Requirement Expression (Alford,Ansart,Hommel,Lamport,Liskov,Mullery, et al., 1985; Mullery, 1979). The CORE method is founded upon the need to produce a requirement specification whose content captures four things.

- Viewpoints. (e.g. Life-Cycle, Environment, Reliability).
- Information Types. (e.g. Actions, Events Mechanisms, Media).
- Relationships. (e.g. Data Flows, Hierarchies, Temporal Order).
- Attributes. (e.g. Size, Accuracy, Frequency, Constraints).

Core serves to define the steps that a problem solver would engage in, in order to produce a requirements specification. For each step CORE carefully defines the objectives and validation checks.

In CORE it is necessary to recognise several viewpoints that may exist e.g. life-cycle, environment, operation etc. For each viewpoint there is then a need to identify the set of information types that may exist e.g. actions, media, events etc. Once the information types have been identified then the relationships that exist between them can be explored along with the attributes that a problem owner would wish to associate with the information types.

The CORE method uses diagrams (Mullery, 1979) to aid in the elicitation and specification of requirements. The diagrams are used to identify the overlapping hierarchical and dynamic relationships that can arise from the many viewpoints inherent in a complete specification of the known requirements. CORE diagrams are produced for both data and actions. They are represented using operational and single thread diagrams respectively. The same diagrammatic notation is used for both of these notations and is illustrated in Figure 2.3.4.1.

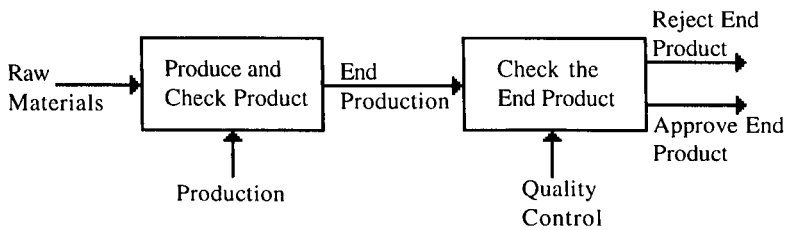


Figure 2.3.4.1 A Core Diagram

The single-thread diagrams represent a form of transaction model and time flows from left to right. The operational diagrams are a time oriented snapshot of the system. The diagrammatical and other notations, i.e. MAL (Modal Action Logic) (Cunningham,Finkelstein,Goldsack,Maibaum, & Potts, 1985), used in CORE can also be mapped down into PSL/PSA (Teichroew & Hershey, 1977; Tse & Pong, 1991) and thus into other more conventional development methods.

The first stage in CORE methodology that a problem solver would engage in is to attempt to define the system boundaries. This activity can by its nature be very problematic. Once this has been achieved, or a preliminary attempt made, then the problem owner would attempt to isolate a small number of top level views of the system. The idèa of multiple view holders of the system is one of the assumptions that CORE is based upon. CORE then proceeds by refining and classifying the viewpoints held by the view holders. At each stage in the refinement process the problem solver would attempt to classify, clarify and decompose the information types, relationships, attributes and functions of the system. At each level of refinement the viewpoints are examined for any contradictions and omissions. If any are found then they are

presented back to the problem owners for clarification. The diagrammatical notation is also used to aid the problem solvers in their quest for the elusive requirements. The problem solver would continue with the refinement and confirmation of viewpoints using the various notations until a structured requirement specification is produced. By making use of techniques to handle conflict identification and resolution the CORE method attempts to guarantee convergence towards an unambiguous problem statement.

The primary purpose of CORE is to aid the problem solver in the production of a requirement specification of the purpose of the system. Once this goal has been achieved CORE then provides the means for this specification to feed into other development methods that could be used to develop and deliver the final system.

CORE views the system from a computational perspective. As a result, it will only allow us to identify, represent and validate requirements from the information and computational components of the ANSA framework (Section 2.2.2). CORE does not let us directly identify, represent and validate requirements from any of the components from the axiological framework (Section 2.2.3).

2.3.5 Jackson Structured Development

Jackson Structured Development, JSD (Cameron, 1986; Ince & Andrews, 1991), is a widely used method for the development of realtime data processing and simulation systems. The technical aspects of the development are divided into three phases.

- The Modelling.
- The Network.
- The Implementation.

The modelling stage forces the problem solver to examine not only the set of informational concepts that the proposed system is to encapsulate, but also the set of entities over which encapsulation will occur. The problem solver also considers what the constraining rules on the concepts are, along with what attributes should be associated with each concept or the manipulation of each concept. The modelling stage is thus used to identify the processes and data that the data processing system is to support. The modelling stage produces a set of actions, entities, attributes and rules. These objects are identified at a very abstract level and later refined. They are then used as input for the network phase. JSD views a system as a set of communicating processes, where each process holds its own local data and the processes communicate

via message passing. The network phase refines the ideas and objects presented in the modelling phase in order to produce a system specification that is then used as input for the implementation phase. In the implementation phase, two main issues are addressed, how to run the processes that comprise the specification and how to store the data that they contain. Thus the implementation stage takes as its input a specification produced by the network stage and produces as its output the final data processing system.

JSD views systems as a set of communicating processes and does not take into consideration how the user's environment will be changed. JSD also does not support the idea of problem owners holding views of the problem and thus of the system. So we may class the JSD methodology as a hard systems methodology.

The JSD methodology does not provide a means by which it is possible to examine the axiological concepts (Section 2.2.3) It does provide a means by which the problem solvers and problem owners may investigate the information and computation projections from the ANSA framework (Section 2.2.2). This ability comes from JSD's skill to view the system as a set of communicating processes through which information flows and is manipulated.

2.3.6 Hierarchical Development Methodology

The HDM, Hierarchical Development Method (Silverberg,Robinson, & Levitt, 1979), is designed to aid the problem solver in the task of requirement elicitation, analysis and specification, and then to aid in the process of design, development and implementation of the proposed information system. The HDM method was designed along the classical software life cycle model (Boehm, 1976). For each stage in the software life cycle model HDM defines a methodology module. Thus the HDM methodology allows a problem solver to plug in at each stage in the development process the particular module that is required.

The specification method should be able to define completely the external behaviour of the components that comprise the information system. The requirement that the specification be precise and unambiguous dictates the use of a specification language. The HDM methodology is very much centred on the specification of the hard information system and thus HDM is a hard systems methodology.

The HDM methodology defines an information system as a set of modules that communicate together via message passing. In order to capture and model this at the specification phase HDM has a specification language called *SPECIAL*, SPECIfication and Assertion Language. The primary purpose of this language is to allow the problem

solver to specify the modules and the communication functions of each module. The specification model of the system at this level may be considered to be an object model of the information system. HDM uses HSL, Hierarchy Specification Language, to describe the structuring of modules into machines and machines into systems. The HSL allows a problem solver to specify quite abstractly the functions that the systems is to perform, and then slowly refine them into a more state transition based form.

HDM views the system from a computational perspective rather than a humanistic one. Consequently, it will only allow us to identify, represent and validate requirements from the information and computational components of the ANSA framework (Section 2.2.2), as it views the system as a set of communicating processes through which information flows and is manipulated. HDM does not let us directly identify, represent and validate requirements from any of the components of the axiological framework (Section 2.2.3). This is due to its view of the system as being a purely computational object.

2.3.7 Forest - Structured Common Sense

The FOREST, (FOrmal REquirements Specification Techniques), project (Cunningham et al., 1985) developed a formal notation, MAL (Modal Action Logic) to allow a problem solver to express, explore and reason about requirements. It also developed a method called Structured Common Sense, (SCS), (Potts,Finkelstien,Aslett, & Booth, 1986). The purpose of this methodology is to aid the problem owner in the task of requirements elicitation and capture. It consists of a number of distinct steps, some of which are performed in parallel and some in series. Each step produces one or more formatted representations and an increment to an evolving formal specification. The formatted representations include a variety of diagrams, tables and highly constrained natural language descriptions. Each step has two aspects: a work plan or strategy, which describes how to do the work involved in the step; and a set of heuristics or tactics, which are hints, tips and general guidance on what to do in particular situations. SCS makes use of other formal notations, i.e. Entity Relationship Diagrams (Chen, 1976), in order to capture requirements. In SCS the formal specification is built up incrementally during the steps rather than being written down as a final step. The final system specification is expressed in MAL. In SCS there are five distinct stages that a problem owner must engage in in order to achieve a final specification. These are:

- Agent Identification
- Data Flow and Action Analysis

- Entity Relationship Attribute Analysis
- Permission and Obligation Analysis
- Temporal Analysis

The agent identification stage can be viewed as the drawing of system boundaries. These boundaries serve to define what lies within the scope of the system. The data flow and action analysis stage determines the links that can exist between the agents and the nature of the interactions that can exist between the agents. It also serves to identify the primary actions that agents may perform along with how these actions may influence each other. The entity relationship attribute analysis stage allows for the identification of the functions that may be performed by an agent, and provides a means by which it is possible to explore the pre and post conditions for the functions. The permission and obligation analysis stage determines the causal forces exerted between actions. The temporal analysis stage determines the relations between actions occupying temporal intervals.

Forest and Structured Common Sense provide a conceptual framework from within which it is possible to explore and examine information and computation components of the ANSA framework (Section 2.2.2) only. This is a result of FOREST's ability to model and analyse data flows and actions. In addition, this method will let us partially examine the axiological concepts and operators (Section 2.2.3) that the organisational information system is required to support. This is due to FOREST's ability to express and analyse permissions and obligations.

2.3.8 Role, Function, Action Nets.

The Role/Function/Action-Net (RFA-Net) method was first presented in (Oberquelle, 1998), and views organisations as communication processes. This method forms part of a larger class of methods interested with Business Process Re-Enginering - BPR (Davenport, 1993). BPR methods are concerned with re-structuring and re-engineering organisations so as to increase their performance. This re-structuring and re-engineering often involves the development, or re-engineering, of the organisational information system.

Role/Function/Action-Nets are a visual framework for the elicitation, representation and validation of requirements. They achieve this goal by the modelling of an organisation at three conceptual levels: the role level, function level and action level.

The role level concentrates on the pragmatics of work. A meaningful task for a person is called a *Role*, the person is called the *Role Player*. A role comprises the responsibility for the task and the rights and duties with respect to other roles and their role players. One Person may play many roles, each role being called a *SubRole* and every role contains at least one function.

On the function level we concentrate on the static organisation of work. A function conceptually comprises one sequential activity and its local resources. All functions are performed by *actors*, each actor may be a person or a machine. A function may access objects and data located in positions. The totality of positions constitutes the organisational space. The local space of a function is called its depot. The space that functions have in common with other functions is called their interface. Functions interact with each other by the exchange of objects or data. An object is considered to be a set of individuals with some attributes and data is considered to be an attribute stored in an object.

The action level concentrates on the dynamics of the work processes. The simplest unit of activity is called an elementary operation. Operations are actively executed by actors and can be used to modify objects or data. The relationship between the operators and the actors is called the control aspect.

The RFA net approach allows a system designer to explore the way and thus the consequences of the way in which people within an organisation interact, how they do their work and what objects and resources have to be present within the system for them. All this aids the system designer in the task of capturing the system requirements.

RFA nets allow for the identification, representation and validation of the information and computation components of the ANSA framework (Section 2.2.2). They allow for partial identification, representation and validation of the enterprise component of the ANSA framework. This ability stems from Role/Action/Function nets functional view of system's behaviour. Role/Action/Function nets only partially allow for the identification, representation and validation of the first component of the axiological framework (Section 2.2.3).

2.3.9 The SAMM Method

The Systematic Activity Modelling Method, SAMM (Lamb,Leck,Peters, & Smith, 1978; Stephens & Tripp, 1978) was developed by the Boeing Computer Services Company. The objective of SAMM was to model an information system using hierarchical decomposition and data flow. The resulting language is a combination of

graphics and graph-theorical notations. The SAMM specification technique is based upon the idea of specifying and decomposing the activities that people within an organisation engage in. A SAMM specification consists of a context tree, a set of activity diagrams and a condition chart.

The context tree is the hierarchical structure that is used to rank activities. It is used to organize the refinement of an activity into its subactivities. It is possible to map from a context tree to an activity diagram quite easily. The context tree allows a system designer to engage in a top down design process, and thus aids the designer in the task of refinement. An activity diagram consists of a description of subactivities and a data table, as shown in Figure 2.3.9.1.

The activity diagram that is shown in Figure 2.3.9.1 depicts the activity (function) of compiling a list of names and then extracting from that list the most common surname and Christian name. The data table is made up of data descriptions with indices and is shown on the left of the activity diagram. In the activity diagram a subactivity is referred to as an activity cell and is represented as a rectangular box. The data flow is drawn as arrows going from one activity cell to another.

Then, having organised the tasks into a context tree, an activity diagram can be used to capture the data flow that exists between the various activities. Once this goal has been achieved the behavioural properties of each activity cell can be specified using a condition chart. The SAMM technique fails to capture the notion of views or projections of a system by implying that the problem is fixed and that everyone shares the same definition of what the problem is.

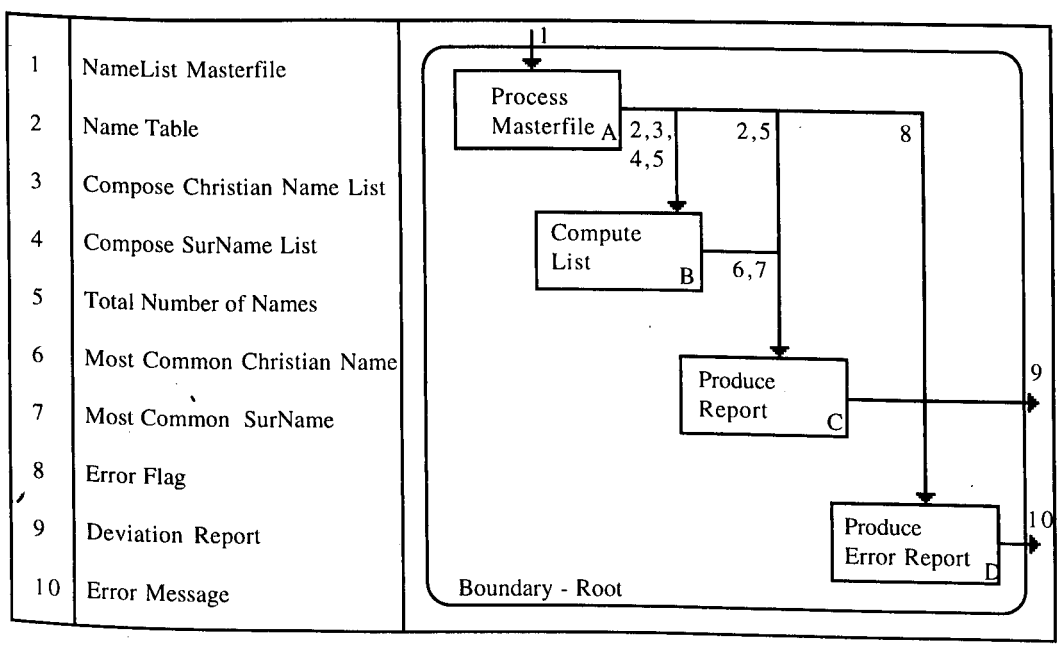


Figure 2.3.9.1 A SAMM Activity Diagram

The SAMM methodology allows the problem solvers to examine and explore the information component, and partially examine and explore the experprise component, of the ANSA framework (Section 2.2.2). This ability is a result of the context tree and activity diagrams that SAMM utilises. The SAMM methodology does not support the examination and analysis of any components of the axiological framework (Section 2.2.3).

2.3.10 Discussion of Methods in General

In this discussion some methods that have already been reviewed will be discussed in addition to some which have not been reviewed. When viewing, analysing and understanding methods it is important to comprehend not only the political concepts embodied within each method, but also the political system within which the method is to be used. It is also important to understand the view that the method has of the world and the problem within that world that the methodology is trying to solve.

Hard systems analysis methods such as JSD (Jackson, 1983) view the computer system as a pure computational object. These methods were developed in, or around the 1960's when computers were big boxes that lived in large rooms and were used mainly as number crunchers. They place emphasis on the information requirements of the system. These methods commonly focus on the flow of information through a given environment and the different entities that make up that environment. They view an information system as an input process connected to a computational process connected to an output process. The goal of these methods is to specify a) the required input, b) the transformations that may be performed upon that input, and c) the output produced by the transformations. Critics make the point that hard systems design methodologies are based on pseudo objective models of systems and suffer from two faults. Firstly the representations used by the analysts only contain what the analyst thinks is important, and secondly analysts tend to fill in gaps in available information using their own intuitive judgement. Hard systems methodologies view the process of problem solving as being deterministic. They also view the process of problem solving as being a clear linear sequence which defines how the problem and solution will evolve and ultimately be defined. These methods on the whole only concentrate on the information and computation components of the ANSA framework (Section 2.2.2). They do not focus on any part of the axiological framework (Section 2.2.3).

The Soft Systems Methodology (SSM) approach to design was initially developed by Checkland (Checkland, 1986). The soft systems methodology can be

seen as a general problem solving approach appropriate to human activity systems. The essence of the approach is an appreciation that for most problems, there are a number of problem statements that may be appropriate, and that the appropriateness of different solutions is largely determined by the particular viewpoints of those people who have an interest in the problem. The soft systems approach provides a conceptual framework to aid in the understanding of how the proposed system will function, along with how the proposed system will be defined. For example, a social security benefits system could be viewed as a system for ensuring that the needy obtain the benefit they deserve, or alternatively it might be viewed as a system for the protection of the "public purse". Soft system thinking differs most from other approaches in that it allows the problem solver to explore the fuzzy and ambiguous nature of a problem. Soft systems methods in general allow us only partially to identify, represent and validate the enterprise and information components of the ANSA framework (Section 2.2.2). We can only partially identify, represent and validate all the components of an axiological framework (Section 2.2.3).

Object oriented analysis and design methods such as (Booch, 1991; Coad & Yourdon, 1991; Rumbaugh,Blaha,Premerlani,Eddy, & Lorensen, 1991), have been applied to both business process re-engineering (Davenport, 1993) and analysis and design of organisational information systems. The methods in general view the system as a set of interacting objects, where the interaction is of the form of strongly typed parameters passing through service invocations. As a direct result they only allow for the identification, representation and validation of the informational and computational components of the ANSA framework (Section 2.2.2). These notations on the whole have been designed by computer scientists for computer scientists and as a result pay no attention to any of the axiological concepts or operators (Section 2.2.3) that an information system would be required to support.

The Scandinavian style of methods can be viewed as empowering the problem owners through education and the free exchange of information. This style of method has been slowly gaining in popularity and is now used in Europe and North America. The ISAC (Avison & Fitzgerald, 1988; Lunberg, 1982) method is a problem oriented method and seeks to identify the fundamental causes of the problem. The approach taken by the ISAC methodology is designed to analyse problem owners' problems and to solve aspects of them where appropriate. The ISAC approach to problem solving is to educate the problem owners by helping them to understand better the nature of their problems. The ISAC method is a stepwise methodology that starts with trying to understand the problems that are facing the problem owners. It then goes on to analyse the information structures that exist within the current system and finally ends with the

system delivery stage. The key difference between ISAC and other methods is that it directly places in the problem owners' hands the task of problem identification and supplies the tools and techniques to achieve this. This class of methods allows for the identification, representation and validation of the second component of the ANSA framework (Section 2.2.2). In general they do not excel at the identification, representation and validation of any of the components from the axiological framework (Section 2.2.3).

Socio–technical systems theory stresses the loss to overall effectiveness of any endeavour which concentrates unduly on technical considerations by excluding or minimising the motives and skills of people who are required to interact with the technology. Human characteristics and needs must be considered in the specification if a system is to be effective from both the technical and human perspective. A number of methods have been developed around the theory that specifically address human issues in design. e.g. ETHICS (Avison & Fitzgerald, 1988), PORGI (Kolf & Oppelland, 1983), OSTA (Harker & Eason, 1985), Pava's Sociotechnical Design Methodology (Pava, 1983) and IT-UPTAKE (Ryan,Wynne,Cullen,Ronayne, & Dolphin, 1988) Characteristically, such methods include specific techniques to assist in identifying the needs and requirements of different classes of users of IT systems, and provide ways of identifying the human implications of design ideas. These methods in general only allow for the identification, representation and validation of the enterprise (and partially the information) components of the ANSA framework (Section 2.2.2). In addition, they only partially allow for the identification, representation and validation of any component from the axiological framework (Section 2.2.3).

2.4 Requirement Specification Languages

2.4.1 PSL/PSA

PSL (Teichroew & Hershey, 1977; Tse & Pong, 1991) (Problem Specification Language) was designed and developed at the University of Michigan in the ISDOS project. PSL was designed to allow a system designer to specify a set of requirements formally and then analyse them automatically (PSA). A PSL statement will cover the system behaviour, system structure, data flows, system dynamics, data structures and project management. The semantics and syntax of PSL is based on the entity relationship approach (Chen, 1976). An example of a PSL specification is given in Figure 2.4.1.1

PROCESS : remove-item ;
DESCRIPTION :

```
        This process removes an item from a list of items. If the list is empty then the
        value zero is returned                                          ;
GENERATES                          : item                               ;
RECEIVES                           : list-of-items                      ;
PART OF                            : list-item-handler                  ;
DERIVES                            : item                               ;
        USING                      : list-of-items                      ;
PROCEDURE                          :
        1.      Check to see if list is empty, if so then place the value zero on the output.
                                                                        ;
        2.      Get the first value on the list and place that on the output.
                                                                        ;
HAPPENS                            : 1 TIMES-PER access                 ;
TRIGGERED BY                       : remove-acss-event                  ;
TERMINATION CAUSES                 : return-item-event                  ;
SECURITY IS                        : all                                ;
```

Figure 2.4.1.1 A PSL Specification

PSL supports a multi-level refinement process, so that systems can be specified in a hierarchical manner. In a PSL specification the logical attributes of the system are separate from the physical attributes of the system. PSL was not designed for a particular system development methodology. It can only be input as a textual language which can then be used to generate a set of graphical reports. There is however no one-to-one mapping between the graphical reports and the textual input. The graphical reports serve to provide a means for presenting the specification back to the problem owners for validation and to aid further the problem owners in their task of requirements engineering.

PSL/PSA was developed to aid in the analysis of the behavioural and functional aspects of the system. As a result it will only allow us to examine issues associated with the information and computation components of the ANSA framework (Section 2.2.2). PSL/PSA does not allow for the examination and classification of any of the axiological concepts associated with the system (Section 2.2.3).

2.4.2 HOS

HOS (Hamilton & Zeldin, 1976; Tse & Pong, 1991) is a requirements specification language that has been developed by Higher Order Software Inc., to support and aid in the automation of the HOS method. The HOS method is a formal method for the specification and development of reliable systems (Hamilton, 1974) and thus is only concerned with the computable functions of the system and their relationships (Hamilton & Zeldin, 1983) e.g. hierarchical decomposition of functions into subfunctions. The purpose of the HOS methodology is to generate provably correct systems designs. In order to do this the formal model is constructed on a set of

mathematical axioms. The system representation is called a control map and is based on a binary tree, see Figure 2.4.2.1.

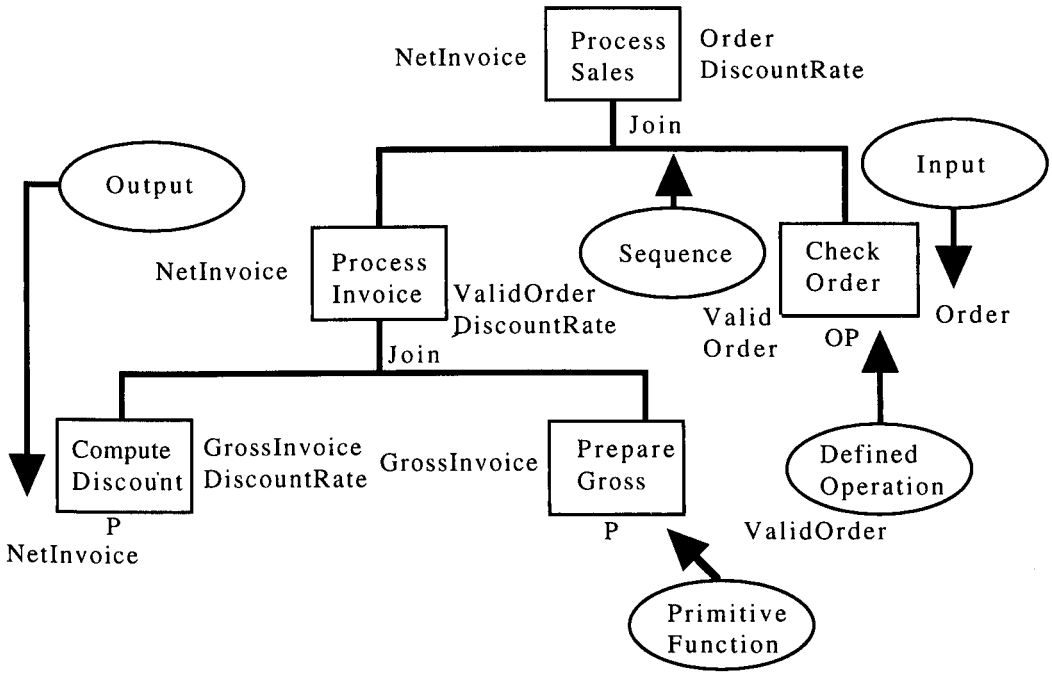


Figure 2.4.2.1 A HOS Specification

Figure 2.4.2.1 depicts the process of producing an invoice for an order. It clearly shows the sub processes and operations that must occur for the process to be completed successfully. Each module in the system is represented by a node in the binary tree, which in turn represents a mathematical function. The input to the module is represented as the domain of the function and the output is represented as the range of the function. The method for decomposing functions into subfunctions is mathematically based and described in the methodology. The subfunctions are represented as children on the binary tree. There are two forms of HOS, a graphical one and a textual one, these two forms are isomorphic and there is a one-to-one mapping from one form to the other. In a HOS specification the domain of the function is displayed on the left of the node and the range of the function is displayed on the right of the node. The type of function is displayed under the node.

The HOS views the world from a computable function perspective and thus the notation facilitates in the comprehension and exploration of only the computation and information components of the ANSA framework (Section 2.2.2). HOS does not allow for the analysis of any components from the axiological framework (Section 2.2.3).

2.4.3 Data Flow Diagrams (DFD)

A data flow diagram, DFD (DeMarco, 1979; Dennis, 1974), is used to describe at a high level how data is transformed as it moves from one system component to another. It documents how data as input is transformed into data as output. In essence data flow diagrams are made up of three components.

- Data Flows (Annotated Arrows)
- Transformations (Annotated Bubbles)
- Logical operators * and ≈

The annotated bubbles represent transformation centres with annotation specifying the transformation. Arrows represent data flow in and out of the transformation centres with the annotations naming the data flow. The two operators are used to mean AND and EXCLUSIVE-OR. The notation allows for decomposition of transformations into sub-transformations. Consequently data flow diagrams promote the principle of hiding unnecessary complexity.

Figure 2.4.3.1 shows two typical payroll functions expressed in data flow notation. The employee time card data and employee payroll data are combined to yield both the new cumulative payroll data and the employee pay data. The employee pay data (or, in a test mode, the simulated data) is then used to compute the pay, which is checked against a master employee list as a safeguard and then the paycheck information (or simulated output for checking) is produced.

It is important to note that a true data flow diagram contains no control flow or sequencing information. One of the principle advantages of data flow diagrams is that they allow the problem solvers to examine how data flows through, and is transformed by, the system without specifying any implementation details.

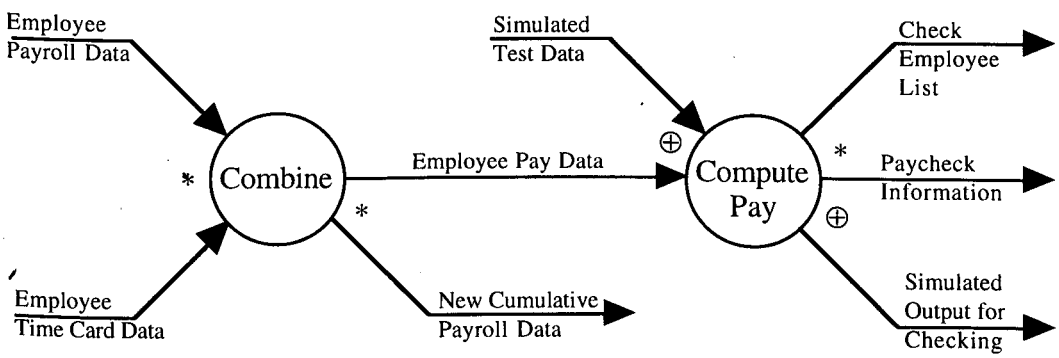


Figure 2.4.3.1 A Data Flow Diagram

DFDs view the world as a series of flows and transformations, thus we can only capture, validate and analyse issues associated with the information and computation components of the ANSA framework (Section 2.2.2). DFDs do not directly allow for the identification, represention and validation of any issues associated with any components of the axiological framework (Section 2.2.3).

2.4.4 Other Formalisms and Languages

There are several other pieces of work that have been done on requirements specification formalisms that are worthy of mention. PAISLey (Zave, 1991) is an executable specification language based on the functional programming language model of computation. PAISLey combines formal representations of both data and control flow and thus the language has a formal semantics. The idea of using Petri nets has also been used to model office information systems and thus capture requirements (Beslmuller, 1988; Rein & Singh, 1992). Petri nets can be used to show how functions are related and thus depend on one another. Using Petri nets we can model and analyse the state changing operations associated with the system. Figure 2.4.4.1 shows how a simple Petri net model of an organisation can be constructed. Through the use of PAISLey and Petri-Nets we can analyse the information flows, computations and transformations associated with the system.

Figure 2.4.4.1 A Petri Net Model

The idea of exploring and specifying a problem has been explored and examined (McLean, 1984) and various notations have been suggested like Z and VDM. These notations however are all built on set theory and first order predicate logic. In (Lee, 1988) a formal system is presented that will allow a system designer to specify a problem. This formalism is built on deontic logic. Deontic logic has two operators, the obliged and permitted operators represented as an **O** and **P** respectively. It is however possible to specify one operator in terms of the other, as shown in figure 2.4.4.2

$$P\Phi \leftrightarrow \neg O \neg \Phi$$

Figure 2.4.4.2 A Example of Deontic Logic Expression

Using deontic logic it is possible to examine some aspects of the way in which people within organisations work. (Lee, 1988) also presents the simple languages DR and PN for the processing of deontic rules; DR is an applied form of a logic programming language and PN is an extension to DR to accommodate the dynamic

aspects of organisations. A system designer using DR may examine the consequences of a particular specification of an organisation.

SXL (Lee & Sluizer, 1991) is a executable modelling language that describes systems behaviour rather than software structure. Using a conventional state transition framework, model behaviour is determined by rules that define pre and post conditions for each transition. Behaviour can also be specified by constraints, logical invariants that are automatically enforced during the execution of the model. The intended purpose of the language is to provide a simple model that directly corresponds to the informal and high level descriptions from which it is derived. Rules and constraints are expressed solely in terms of entity relationship structure and declarative logic. The language lacks machine oriented data and control structures and has no facilities for specifying or implementing software. A system designer can thus capture requirements about the systems behaviour without making any design decisions.

Diplans (Holt, 1988) are the expression of a graphical language used to describe plans of operation in human organisations. Just like any other task co-ordination takes effort. Yet it is upon co–ordination that most organisations are built. What a diplan attempts to do is to model the way in which people interact with each other and their relationships with each other. Another model of co–operation is that of Interconnected Roles (Singh, 1992). In this notation the concept of role is mixed with that of Petri nets to provide an executable model of how organisations function. In (Singh, 1992) a role is viewed from a functional perspective as a precise functional specification of a piece of behaviour. Both Diplans (Holt, 1988) and Interconnected Roles (Singh, 1992) allow us to model and analyse information flows.

A notation for the modelling of organisations using speech acts is presented in (Lyytinen,Auramaki, & Hirschheim, 1991) along with a method SAMPO (Speech Act based office Modelling aPprOach) to aid in the construction of the model. The idea of speech acts was first described in (Austin, 1962) and later developed in (Searle, 1969). In (Auramaki,Lehtinen, & Lyytinen, 1988)a graphical front end is added, this allows a systems designer to explore how, and in what way the people within an organisation interact. It is possible to examine how commitments are established within and flow through the organisation, as well as to examine what objects act to signal the establishment and discharge of commitments. These organisational modelling techniques suffer from being too complicated to use and too difficult to understand. Notations such as (Lyytinen et al., 1991) allow us to examine issues associated with the information and computation components of the ANSA framework (See Section 2.2.2). They also only allow us to examine logical operators associated with the axiological framework (See Section 2.2.3).

2.4.5 Discussion

Researchers in the area of software engineering have for a long time used formal methods as a means of specifying systems. The belief is that if the system is specified correctly using certain logics then various theorems about the properties of the system can be tested. In addition, a validation function may be performed on the system to check that the developed system meets the specification. The problems that have confronted formal logic researchers have primarily been concerned with how a correct specification is constructed. Specification languages such as Larch (Guttag,Horning, & Modet, 1990), Z (Spivey, 1989), VDM (Lucas, 1987), OBJ and CLEAR (Goguen & Burstall, 1977) all draw upon various logics to aid in the construction of a specification. By considering purely functional aspects of the system they fail to capture correctly the context and thereby the environment of the system. In limiting their view of the world to a purely functional one they miss all of the nonfunctional information that is so important in defining todays systems.

None of above, however, provide any type of methodological framework to aid in the process of specification construction. The problem facing many people using these notations is one of constructing and validating a specification that accurately reflects the needs and values of the system owners and system users.

Various diagrammatic notations have been developed to aid in the requirements engineering process. These range from informal notations like Role/Function/Action-Nets (Oberquelle, 1998) and SAMPO (Auramaki et al., 1988) that draw upon social concepts to aid in the construction of a specification, to the more formal notations like Petri-nets (Beslmuller, 1988) and Role Interaction Nets (Rein & Singh, 1992) that draw upon formal systems to assist in the construction of a specification. All of the above have failed to address the issue of cognition of usage. They all present complex diagrammatic notations that draw upon various formal notations, and which are are very difficult for an untrained and unskilled person to pick up and use.

Chapter 3 *The Architectural Perspective*

3.1 Introduction

In (Wittgenstein, 1958) the concepts of what a language is and how it is used are carefully explored and examined. In addition, the idea of what words like *game* mean, and how we might try to define them are investigated and their implications explored. We may define the syntax of a language or the formal rules of a game, however it is the usage of a language and the playing of a game that gives them their meaning. The problem with modelling words like responsibility, obligations, policy and information is that while they occur, and are used, in everyday life, they are words that cannot be defined except when placed in context. This analysis led (Moore, 1993) to conclude that "it is through the process of establishing a shared language that a set of shared values, ethics and interpretations are captured".

Previous methods and notations, such as JSD, CORE and SSADM, have all tried and failed (to greater or lesser extents) to model words such as obligations and responsibility so as to elicit and comprehend organisational requirements and change. The reason for this failure is that they have all attempted to model such words in an absolute and unitary sense, rather than attempting to capture their context and usage. The term absolute is used to denote a definitive statement that serves to define the precise meaning, while the term unitary denotes that the word is used to describe an atomic building block rather than a relationship. This absolute definition approach stems from traditional hard systems thinking, where a system is viewed as purely a computational object. The semantics of such systems are assumed to be formal in nature and the implication is that once enough state information has been captured, then the whole system may be modelled and understood. The similarities between the hard systems thinking approach and Newtonian mechanics are significant.

No modelling approach can avoid philosophical assumptions because modelling is a process of inquiry that has intrinsic similarities with classical scientific theory construction (Klein & Hirschheim, 1987). The analysis of such assumptions classifies them into one of two sets: ontological or epistemological. When modelling systems within an ontological framework, two ontological positions emerge: realism and nominalism. Realism postulates that the universe is made up of immutable objects and structures. In contrast nominalism asserts that socially transmitted ideas and names direct how reality is perceived and structured. When modelling systems within an epistemological framework, two epistemological positions emerge: positivism and interpretivism. Positivism asserts that world can be explained through the identification of causal relationships. In contrast interpretivism postulates that the understanding of the world is mediated through social relationships.

Methods and notations such as JSD can be classified according to the above definitions. In each case they would tend towards the realism and positivism end of the spectrum, with JSD firmly embracing them both. However to embrace fully both realism and positivism leads to a rigid view of the world; methods of this type are called hard systems methodologies. Newtonian mechanics, and methodologies and notations such as JSD have taught us that to attempt to model and comprehend complex systems on a realistic and positivistic level is subject to certain limitations. I have rejected the views of realism and positivism in favour of nominalism and interpretivism. The reason for this stems from the following quote "*Two things have influenced organisations more than anything else, the coffee machine and the computer - Why ? because they mediate social relationships*" (Ehn, 1989). There is growing evidence to suggest that adopting a realism and positivism approach to the specification and design of organisational information systems does not work, and that adopting a nominalism and interpretivism approach does (Wiener, 1993). Consequently I have adopted a more comprehensive, sociological, contextual and environmentally driven approach.

An organisation may be viewed as a dynamic set of social relationships that are in a constant state of flux. As problem owners will form an integral part of the organisation any methodology or notation must obviously take into account their perceptions of what the problem is. This generates the requirement on a methodology that it be a fluid entity which can be moulded to embrace any of the problem owner's perceptions. The philosophy behind this approach is one of *problem owner involvement at all levels* and *the method and notation working together as a medium of communication to achieve and build a consensus*. This is in contrast to other more traditional methods or notations that impose the use of a particular political, sociological or technical perspective.

3.2 What is an Architecture

Over the centuries numerous architectures have emerged to represent the various periods in history, as for example Roman, Norman, Gothic, Tudor, Victorian and Post-Modern. Each of these has not only represented a particular set of design choices and ethics but also encapsulated a certain cultural view of the world and mankind's relationship within it. The term *architecture* is generally used within the computer system community to imply some kind of formal framework, or rationale, within which it is possible to define and reason about computer systems.

The notion of an architecture as a means of specifying, analysing and validating systems has been presented in (Zachman, 1987) and further developed in (Zachman & Sowa, 1992). The problem with many current software development processes and support systems is that they have an inbuilt and implicit architecture that is forced upon a software engineer when attempting to use that process (Perry & Wolf, 1992). An architecture does not simply consist of a set of basic building blocks, a set of prefabrications, a set of styles and a set of techniques and heuristics, but has in addition technical, social, organisational and political concepts. For example, when we examine Norman architecture we can clearly see a set of prefabrications of things such as windows and doors, we can also observe a lot of ornamental work. This ornamental work in fact reflects the Norman's perspective of the relationship to the world around them. Consequently we may say that the choice of an architecture is a cultural choice and therefore a socio-political statement. It follows that any software engineer attempting to utilise that particular architecture should be aware of the implicit concepts and their implications so as befittingly to apply the architecture to the problem domain.

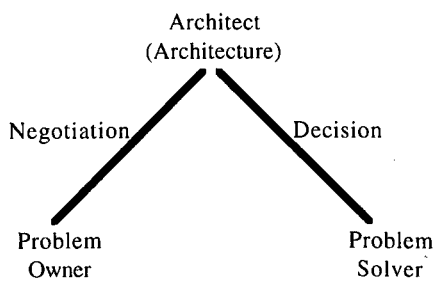


Figure 3.2.1 The Purpose of an Architecture

There is more to the use of an architecture than a set of components, rules and guidelines used during the process of system specification. An architecture is also used as part of the process of negotiation about what sort of system is required (Alexander, 1979) and as part of a decision process about the system that should be built (See

Figure 3.2.1). The negotiation process can only be successful if all parties involved share a common language.

The architecture can be used to express, explore and reason about both the problem and the solution aspects in both the social and technical domains and may consequently be used to build consensus. From the simple building blocks and modelling language a set of more complex and structured models and prefabrications can be constructed and reasoned about. A prefabrication is a template that can be instantiated, or configured, to yield an object or design. In a particular architecture we may have a prefabrication for a window or a door, the architect then instantiates and configures these components to meet the requirements. Within this thesis we use a prefabrication in the same way.

In the enterprise and information projections a prefabrication is an object over which we have a shared understanding, a familiarity of usage and a shared set of common semantics. It is this breadth of applicability, which by the construction of alternative models allows for the exploration of possible futures, that gives the architecture its expressive powers. The architecture may either be used prescriptively, descriptively or normatively to construct and reason about organisations and their information systems.

The language is used to define a set of simple building blocks from which a set of models and prefabrications can be constructed. The basic set of language components, rather than drawing upon formal denotational concepts, draws upon social and linguistic constructs (for example roles, responsibilities and obligations, conversations, system boundaries) the semantics of which are drawn from the environmental and contextual settings within which they are set. The prefabrications defined can be further elucidated by the problem owners during the negotiation phase. They may be used either to construct and validate a set of models (where each model represents a particular viewpoint or perspective on the problem), or to create a further set of prefabrications. If the components and prefabrications can be defined not only by the problem solvers but also by the problem owners then this will guarantee the applicability of the architecture to a particular problem and solution domain.

The process developed in this thesis must reflect the dual role of any architecture: as a part of the process of negotiation between architect and client, and as part of the process of decision between architect and builder.

3.3 The Enterprise and Information Architectures

An essential part of creating a set of architectural models is the understanding of how these processes of negotiation and decision making relate. The key to this lies both in the constructs of the architecture and in the rules for decomposition and composition. This thesis addresses these issues in the construction of an architecture for the negotiation and specification of systems to support the specification, analysis and validation of organisational and information systems requirements for Information Technology. Organisations are often modelled in terms of activities (e.g. the soft systems approach (Checkland, 1986)). This approach I have rejected, since it too easily leads to systems unfit for their purpose in terms of supporting people in their jobs. I believe that an analysis of why this is so would lead us to the conclusion that it is the (de)composition rules for activities that are at fault for this purpose. Instead, I have defined enterprise and information projection architectures with both composition and decomposition rules based on the axiological concepts of responsibilities and contracts respectively. I will demonstrate in the case studies contained in Chapter 7 and Chapter 8 that the architectures are more applicable to the specification, analysis and validation of organisational and information systems requirements for Information Technology than more traditional approaches.

3.3.1 Components of the Architectures

One of the primary purposes of an information technology architecture is to facilitate the expression and analysis of the organisational cultural aspects and values that the system is required to embody. An additional purpose of the architecture is to provide a medium through which consensus can be built and information and understanding may flow (Zachman, 1987). These goals are achieved through the use of semiotic and hermeneutic components that allow for the expressing of, and reasoning about, the social and technical issues and concerns of the system. Semiotics is the study of signs and how they are manipulated, while hermeneutics is the study of how signs acquire meaning and are usage.

The semiotic components of the architecture are derived from a simple, yet powerful and expressive, modelling language and its vocabulary. The construction of the graphical notation is directly derived from the technical and social domain applicability of the modelling language. The notation has the ability to be used in a descriptive, prescriptive and normative manner. The graphical notation is used to construct a set of models, templates and prefabrications from which organisational requirements may be directly derived and reasoned about. The notation and rules

governing the fabrication of the models have been constructed in such a way so as to facilitate the problem solvers, and problem owners, in their task of capturing and specifying the various components and values which make-up and comprise the organisational information technology system.

The hermeneutic aspects of the architecture are concerned with what, how, when and where the axiological aspects of the domain of problem owners and of problems solvers are embodied in it. The question is what cultural and political aspects and values is the enterprise architecture required to embody? Various architectures may be said to embody and consequently to represent the culture and politics of the time. The hermeneutic powers of the architecture may be said to derive directly from the ability of the problem owners and problem solvers to enshrine their own values, ethics and ethos into its semiotics. The question becomes one of "How, what, when and where are the correct values, ethics and ethos to be found and encapsulated and what do we mean by the term correct?" The answer to question lies in the application of enterprise and information conceptual frameworks (See Figure 3.4.2.1 and Figure 3.4.3.1). The purpose of the frameworks is to: a) define and establish a shared language, and b) structure and direct questions concerning the establishment of a shared language.

3.3.2 The Enterprise and Information Architectures

The modelling perspective that has been adopted for the purpose of this thesis is that of the five ANSA projections. Each of these projections attempts to provide a useful set of concepts so as to better elucidate a particular perspective of open distributed systems via the definition of an architecture. The architectures are defined to consist of an hermeneutic component called a conceptual framework (See Figure 3.4.2.1 and Figure 3.4.3.1) and a semiotic component called the modelling framework (See Chapter 5 and Chapter 6). From the perspective of this thesis the enterprise and information ANSA projections will be defined and further elucidated. In addition the nature of the relationship, and the mapping, between the projections will be explored and defined.

What is new about the conceptual frameworks is that they allow for the political, cultural and technical aspects of the system to be explored and defined. This is a very different approach when compared to more traditional methodologies which have always allowed for only technical aspects of the system to be examined and defined. The term political is used to define the nature of the organisational environment within which the system is required to function. By drawing on the discipline of sociology we

may observe that no aspect of an organisation may be considered in isolation as they all act in such a way so as to influence each other in complex ways.

The enterprise projection architecture addresses issues concerned with the flow and evolution of axiological concepts such as responsibility and obligations. In addition this architecture allows for the delineation and examination of the numerous objects that are used as a result of the evolution of various attributes. The key difference between this and other more traditional notations and methodologies, is that it attempts to elicit organisational requirements with the use of such words in a soft, more flexible sense. In addition the semantics and subtyping of such words, and how they relate to each other, is left to the problem owners and problem solvers to explore and define. It is this socio-technical approach that gives this technique its strength and breadth of domain applicability.

The information projection architecture addresses issues concerned with the creation and manipulation of information. The purpose of this modelling is so that the following questions can be answered, "Can this information processing and manipulating be computerised and what information processing and manipulations are required to support the organisation in the achievement and fulfilment of its goals and policies?" This architecture attempts to use the term information in a more sociological context driven way rather than in any absolute precise manner. It addresses how information might flow through an organisation and is manipulated, created and destroyed. From within the information projection conceptual framework it is possible to explore and examine the various aspects of the information system ranging from axiological to temporal.

3.4 The Conceptual Frameworks

3.4.1 Introduction

The purpose of the frameworks is to: a) define and establish a shared language, and b) structure and direct questions concerning the establishment of a shared language. It is though the process of establishing a shared language that a set of shared values, ethics and ethos are captured (Moore, 1993). So for example, the conceptual framework depicted in Figure 3.4.2.1 tells us that when we are examining responsibilities we also need to talk about and establish consensus over the agency which hold the responsibilities, the rights which the agency needs in order to fulfill them responsibility, and exactly what the responsibilities are.

3.4.2 The Enterprise Conceptual Framework

The conceptual framework depicted in Figure 3.4.2.1 identifies and depicts how the concepts of agency, role and agent are used within the enterprise projection. Figure 3.4.2.1 does not imply an ordering of steps, rather it shows how the concepts are related to one another, and that it is possible to move from one concept to the next. This framework provides us with a conceptual tool with which it is possible to delineate and classify organisational informational policy statements. For example, we could classify and analyse the following policy statement: "Only blue people can hold pink responsibilities." Each concept shown in Figure 3.4.2.1 exists within the enterprise modelling language as a unique component.

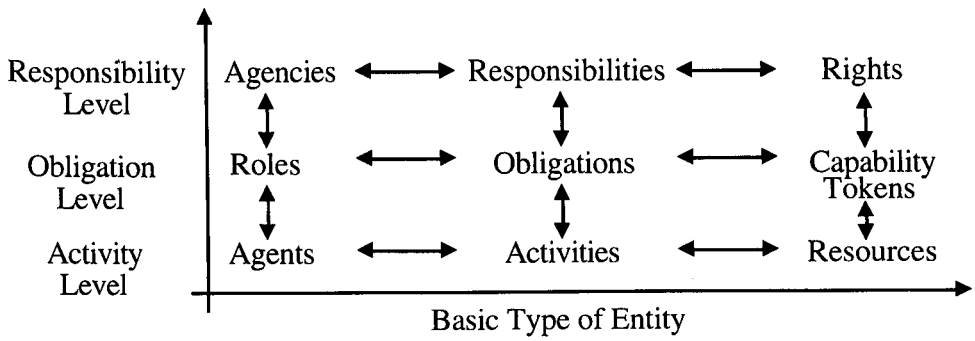


Figure 3.4.2.1 The Enterprise Projection Conceptual Framework

In Figure 3.4.2.1 the rows are defined to represent different aspects or perspectives of the organisational enterprise system. These aspects or perspectives should be regarded as separate but inter-related models. The columns are defined to represent objects or relationships that are of the same basic type but at different levels of organisational enterprise conceptualisation.

Using this framework we may choose to describe and analyse the organisational enterprise system at one of three levels. Each level of the conceptual framework represents a different degree of conceptualisation from the set of activities that make up the organisational enterprise reality. In the activity level we use terms such as agents, activities and resources. At this level the agents are defined to be the manipulators of the system; the activities are the methods by which the system state is changed and the resources, along with the set of relationships which binds all the terms together, as the entities that define the system state. At the obligation level we can look at the organistion in terms of roles performed. In the obligation level we use terms such as roles, obligations and capability tokens. In the responsibility level the terms of responsibility, agency and rights are used. Agencies act as repositories for the responsibilities, thus an agency may be said to hold certain responsibilities. The rights

are concerned with the accessing of the various resources that are required by the agency in order for it to fulfill its responsibilities.

In essence the conceptual framework describe the why of the organisational enterprise process in terms of responsibilities, the what of the organisational enterprise process in terms of obligations and the how of the organisational enterprise process in terms of activities. The conceptual framework functions by providing a set of modelling language constructs with which it is possible to fabricate models of, and reason about, the organisation enterprise system at the given level of conceptualisation.

3.4.3 The Information Conceptual Framework

The conceptual framework depicts how the concepts of contractor, commitment and party are used within the information projection. Each concept exists within the information modelling language as a unique component. Figure 3.4.3.1 does not imply an ordering of steps, rather it shows how the concepts are related to one another, and that it is possible to move from one concept to the next. This framework provides us with a conceptual tool with which it is possible to delineate and classify organisational informational policy statements.

In Figure 3.4.3.1 the rows are defined to represent different aspects or perspectives of the organisational information system. These aspects or perspectives should be regarded as separate but inter-related models. The columns are defined to represent objects or relationships that are of the same basic type but at different levels of organisational information conceptualisation.

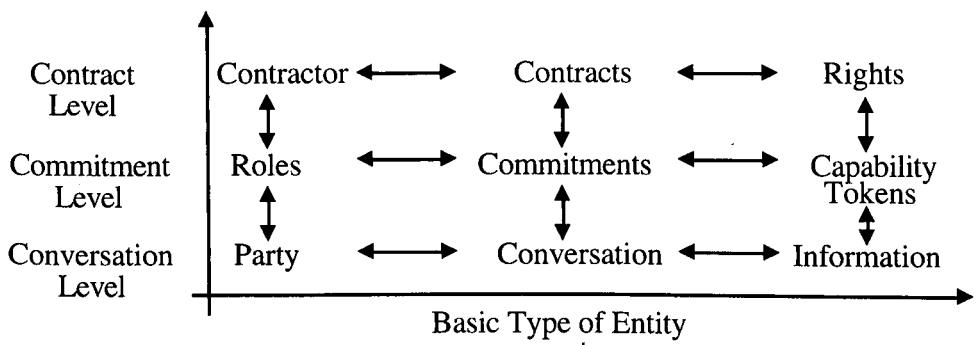


Figure 3.4.3.1 The Information Projection Conceptual Framework

The three rows, or levels, of the conceptual framework are called (from top to bottom) the contract level, the commitment level and the conversation level. Each level represents a different degree of conceptualisation from the set of conversations which make up the organisational information reality. In the most conceptualised model we have contractors, contracts and rights, while in the model of the basic organisational

informational reality we have parties, conversations and information and at the intermediate level we have roles, commitments and capability tokens. In the column on the far right the access rights and capability tokens are concerned with the accessing and manipulation of information.

Using this framework we may choose to describe and analyse the organisational information system at one of three levels. At the conversational level we use terms like parties, conversations and information. In this level the parties are defined to be the manipulators of the system, the conversations are the methods by which the system state is changed and the information, along with the set of relationships that bind all the terms together, as the entities that define the system state. At the commitment level we use terms like roles, commitments and capability tokens. At this level we can examine what the role entails in terms of the commitments that the role holder is required to fulfill and whether the necessary capability tokens have been allocated so that the role holder may fulfill these commitments. In the contractual level the terms of contractor, contracts and rights are used. Contractors function in such a way so as to act as the repository for the required contracts, thus a contractor may be said to define a particular set of contracts. The rights concern the accessing of the information that is required by the contractor in order for it to complete its contract.

In essence the conceptual framework describe the why of the organisational informational process in terms of contracts, the what of the organisational enterprise process in terms of commitments and the how of the organisational enterprise process in terms of conversations. The conceptual framework functions by providing a set of modelling language constructs with which it is possible to fabricate models of, and reason about, the organisational information system at the given level of conceptualisation.

3.5 Mapping Between Projections

The purpose of the mapping process is a) to detect and correct errors, omissions or misconceptions, b) to maximize the data already collected, and c) to further develop and test the validity of the shared language. These are achieved through the use of the process model depicted in Figure 3.5.1. The process begins with the problem solvers and problem owners generating a representation of the problem in a given projection and using the architecture of that projection. This phase of the process model is called "Generate Representation". For a detailed description of the process which results in the generation of a representation of the problem in a given projection the reader is referred to (Chapter 4).

We map between the enterprise and information projections through the use of the conceptual frameworks presented in Figure 3.4.2.1 and Figure 3.4.3.1. Each object in the enterprise projection conceptual framework has a unique counter part in the information projection conceptual framework. So for example, when mapping an enterprise projection diagram into an information projection diagram all the agencies in that diagram are converted to parties and all the obligations are converted to commitments.

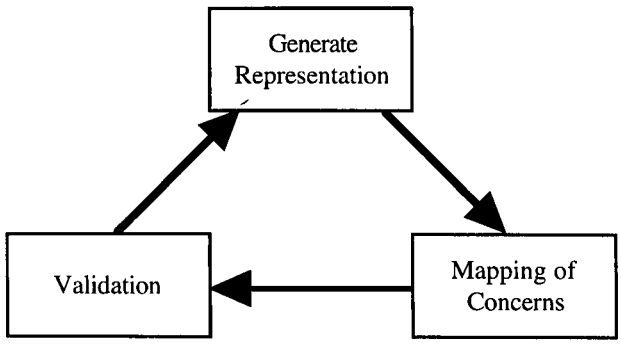


Figure 3.5.1 The Mapping Process Model

The mapping between the enterprise and information projections exists at the syntactic and semantic levels. When mapping at the syntactic level we use the conceptual frameworks depicted in Figure 3.4.2.1 and Figure 3.4.3.1. Thus at the syntactic level the mapping is one to one with agencies/agents being mapped into parties, structural roles and relationships into contractual roles and relationships and finally, functional roles and relationships into conversational roles and relationships. When mapping at the semantic level we have to examine the nature of the roles and relationships and the associated invocation of conversations. Consequently the contractual relationships are derived by comparing the nature of the structural relationships and their associated invocation rights with the various contractual relationships defined in Section 5.8.4 of Chapter 5. For example, when mapping a structural role into a contractual role we ask the question "What state of affairs is this structural role concerned with and how is the conversation associated with this structural role invoked?" From the answer to this question we derive a particular type of service that defines a particular type of contractual role by comparing the answer to the questions to the contractual roles defined in Section 5.8.4 of Chapter 5. This mapping is not automatic and due to the social nature of the relationships and roles involved can not be automated. This mapping has to be done by hand and then verified by the problem owners. This phase of the process model is called "Mapping of Concerns" and is purely syntactic and semiotic in nature. The result of this phase in the process model is a new representation of the problem in the new projection. This representation is then shown to the problem owner and the following question is asked "Does this new

representation adequately capture and express the issues and concerns that it should and that you want it to, and if not, why not ?" The answer to this question is not trivial as it is in essence concerned with the examination and classification of the representation at a cultural and hermeneutic level.

The problem solvers and problems owners then engage in a negotiation process in an attempt to answer that question. This phase of the process model is called "Validation" and may have one of two results. Either the new representation is the projection that is agreed upon, or the feedback loop is established into the generate representation phase and a new representation is constructed. This feedback loop may result in new technical or cultural definitions being established.

The problem owner forms an integrated part of the process, as in this process the problem solver is performing the mapping and representing it back to the user in an interactive manner. It is this constant feedback loop that is used to generate useful insights into the problem owner's perception of the problem and the problem solver's perception of the solution. As a culture is a social construct generated by a group of people, its concepts and process may not be formally defined or represented. The strength of the process model as a whole lies in the cultural mappings which are performed by both the problem owners and problem solvers, and in the interactive nature of the process itself.

3.6 An Example

3.6.1 The Problem Statement

In Chapter 8 a detailed description of the Accident and Emergency department of the given. In this and the following sections I will use only part of the problem to explore the mapping between the enterprise and the information projection.

When constructing enterprise and information models we can identify and validate a set of stakeholders. Here a stakeholder is defined as all those claimants inside and outside an organization who have a vested interest in decisions faced by the organization in adopting and utilizing information technology (Mason & Mitroff, 1981). In addition, we are seeking to build up a shared picture of what the problem is. The establishment and validation of this shared picture is achieved through the creation and application of a shared language.

The easiest way to illustrate the mapping and verification process is with an example. The purpose of this example is to illustrate the ease with which it is possible

to map concerns between the various projections. In the Jervis Street case study described in Chapter 8, at the enterprise projection level the doctor has a relationship with the junior nurse of type service. The doctor must request that the nurse perform a function and due to the nature of the organisation and various legal aspects the nurse may refuse. The meaning of this relationship is that the doctor does not have the authority to order the nurse to deliver a service. What makes this relationship interesting to model is that the doctor first defines what is ailing the patient and then what cure is required. This statement acts in such a way so as to define the service that the nurse is required to deliver. The doctor then formally negotiates with the nurse for the delivery of the specified service. The service that is negotiated for is however delivered to a third party, that of patient, which is excluded from the negotiation process.

3.6.2 The Enterprise View of the Problem

Figure 3.6.2.1 is an enterprise diagram concentrating on the structural aspects of the organisation. In this diagram we can see that the doctor has a *Diagnostician–Client* relationship with the patient. This relationship tells us that there already is a responsibility upon the accident and emergency department to treat the patient. This responsibility is created and monitored by the regional healthcare authorities. Using the conceptual framework in Figure 3.4.2.1 we can ask the following questions to all the stakeholders, "What exactly is the responsibility ?", "To whom does it bind ?", and "What access rights exist concerning the fulfillment of this responsibility ?" The reason for asking these questions is so that we can build up and validate a shared language and a shared picture of what is going on. If the patient believes that the responsibility has not been properly fulfilled then there are a set of formal steps which may be invoked to resolve the situation.

In the *Diagnostician–Client* relationship depicted in Figure 3.6.2.1 the doctor defines what healthcare is required by the patient in order for the accident and emergency department to fulfill its responsibilities of delivery of primary health care by performing a diagnosis. The relationship defines that the doctor is responsible for performing a diagnosis on the patient. This diagnosis is then recorded and forms the specification of the service which is required by the patient and that the nurse is required to render. Using the conceptual framework in Figure 3.4.2.1 we can ask the following question "What access rights exist over the record of diagnosis ?" The doctor however cannot simply order the nurse to deliver the service. Due to various legal aspects of the system, the doctor is required to negotiate with the nurse for the delivery of a service to the patient. Should these negotiations break down then the hospital has defined a set of formal procedures, the function of which is to resolve the situation.

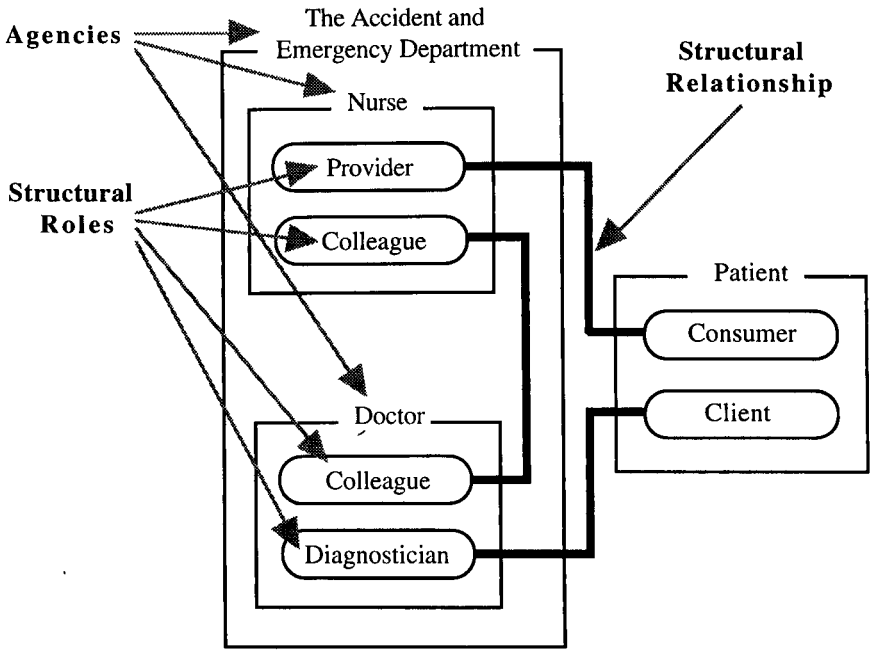


Figure 3.6.2.1 The Enterprise Model

In addition, we can see that the nurse has a *Provider-Consumer* relationship with the patient. In this relationship the nurse defines how the service which fulfills the accident and emergency department's responsibilities to the patient and which was defined by the doctor will be delivered, and delivers it. Again, using the conceptual framework we can ask "What access rights to resources exist as a result of this responsibility ?" The *colleague–colleague* relationship between the doctor and the nurse defines the negotiation process by which the doctor has the nurse deliver the defined service to the patient. The purpose of the structural relationship depicted in Figure 3.6.1.1 is to define the responsibilities that exist between the various agencies. All of the structural relationships serve to define how, when and where the responsibility of primary health care delivery is fulfilled.

3.6.3 The Information View of the Problem

The types of relationships that can exist within the information projection are determined by a contractual view of information. The initial mapping is achieved through the use of the conceptual frameworks (Figure 3.4.2.1 and Figure 3.4.3.1). The frameworks are superimposed ontop of each other. The mapping is achieved by taking the concept that occupies the same space in the conceptual frameworks as the concept you are mapping from. For example, agencies can be mapped to contractors and vice versa. The first thing to note about Figure 3.6.3.1 is that while the agencies have change to parties, and the structural roles and relationships have changed to contractual roles and relationships the notation used to depict them is the same.

At the syntactic level the agency called Doctor that was depicted in Figure 3.6.2.1 is now depicted as the party called Doctor. In addition, the structural roles and relationships that existed in the nurse in Figure 3.6.2.1 have been mapped into contractual roles and relationships (See Figure Figure 3.6.3.1). While at the syntactic level we can see that structural roles get mapped into a contractual roles, at the semantic level we now have to map the types of the structural roles and relationships into types of contractual roles and relationships

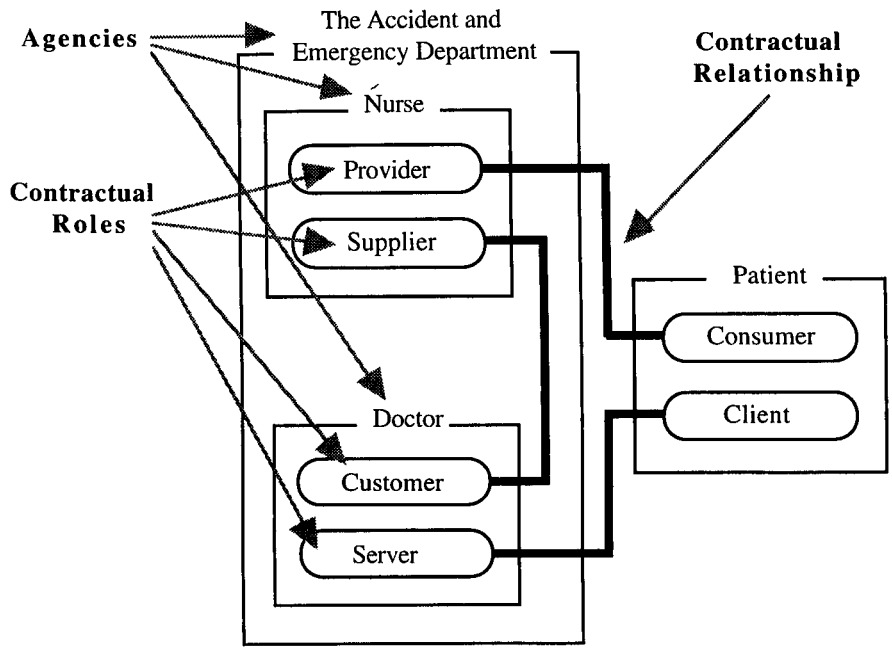


Figure 3.6.3.1 The Information Model

In the enterprise model (Figure 3.6.2.1) the doctor is responsible for performing a diagnosis on the patient. Within the information model this diagnosis is perceived as a service. Due to the fact that this service can be invoked at any time, we map the *Diagnostician–Client* relationship in the enterprise model into a *Client–Server* relationship in the information model. For example, a patient may request the delivery of primary health care simply by walking into the accident and emergency unit and asking for help, while if someone is brought into the unit unconscious then the doctor may simply deliver primary health care. The objective of the service that the doctor delivers to the patient is the production of a diagnosis. This diagnosis acts so as to define the service that the nurse is required, via a negotiation process, to deliver. In addition, in an expanded information model the relationship would also function to define information objects and the accessing rights that are required in order for the successful delivery of the service. Using the conceptual framework 3.4.3.1 we can ask the question "What are the access rights concerning the contract for the delivery of

service by the nurse ?" When answering this question it is interesting to note that the patient does not have an access right to the diagnosis.

In the enterprise model the relationship between the doctor and the nurse was that of colleague, that is to say that the nurse is not responsible to the doctor for anything. In the information model we map the negotiation process between the doctor and the nurse into a *Customer–Supplier* relationship. The reason for this is that the doctor has to negotiate with the nurse to arrange for the delivery of a predefined service. The negotiation process is for the delivery of the primary health care service. Within this process, the diagnosis component of the patient's records acts to define the required service. This relationship defines what access rights are required to the records and how the information contained within the records is to be used. Again, using the conceptual framework 3.4.3.1 we can ask the question "What access rights does the nurse have over the information that is required in order for the nurse to fulfill the contract and deliver the service ?"

3.6.4 Conclusions of the Example

This example serves to demonstrate the nature of the mappings from one set of projection concerns and concepts into another. The key to this mapping lies in examining, understanding and classifying the nature of the relationships that bind the various entities together. This mapping is done at both the syntactic and sematic levels. From the perspective of the nurse the key concern in the doctor—nurse relationship is that the nature of the relationship is one of negotiation and not servitude. The doctor has to negotiate with the nurse in order to arrange for the delivery of a specified service to a third party. Each projection takes this description of the relationship and interprets it within its own conceptual framework. This interpretation leads to an expression of the relationship in terms of concepts present in the projection. Analysis may be performed upon this expression in order to derive requirements.

Chapter 4 *The Method*

4.1 Introduction

Socio-technical systems thinking views the people within the organisation and how they interact with the computer as part of the system. In addition, it views the understanding of the social value system, and the organisational goals and policies that the computer system is required to support as vital to the success of any procurement, development or utilisation of an information technology system. In short socio-technical systems concentrates on defining an information technology system from a human perspective. Technical systems thinking (or hard systems thinking) views the information technology system as a purely computational object, and the discipline of the development of such systems as purely an engineering discipline.

When designing complex systems the problem of handling complexity can arise at two levels: 'technical' and 'organisational'. Engineers have built up an extensive body of knowledge that enables us to deal with technical complexity. There is no substantial body of knowledge that enables us to deal satisfactoril with organisational complexity. The traditional notion of the software development life-cycle with requirements capture being completed before the design stage is no longer satisfactory and is now considered harmful (Fagan, 1974; McCracken & Jackson, 1982). Studies have also been performed on the performance and failure of methodologies that adopt the traditional life-cycle view as their underlying model of development (Boehm, 1976; Daly, 1977; Fagan, 1974). Requirements capture and design are now seen to be symbiotic. The initial set of requirements needed to start off the design process is gradually extended into a systematic and coherent statement of requirements hand in hand with the refinement of the design.

Within organisations, large tasks tend to be devolved to groups of people who work together in complex ways to achieve an overall objective. This has always been the case, and yet technical systems design tends to assume a single user with a discrete task. The failure to recognize that users work in a collaborative or co-operative way, and to design systems to support this way of working, can account for the relatively

low success rates of many complex technical systems. One of the aims of the technique that will be presented in this thesis is therefore to enable design teams to address these organisational requirements, and thereby to produce IT systems that match not only the organisational and functional needs of the individual end user, but also those of groups of users and their associated usability and acceptability requirements.

In order to achieve the aim of supporting different members of the design team the method must:

- identify the full range of relevant requirements in a specific organisational context;
- derive appropriate functional specifications for any IT systems which take account of organisational as well as individual task requirements;
- compare the organisational requirements match of different design alternatives;
- represent the range of organisational requirements to the problem owner and the systems designers in an iterative way;
- identify organisational mechanisms or processes for fulfilling critical non-functional requirements.

4.2 The Method

4.2.1 Introduction

The method used to determine requirements should allow the problem solvers to explore possible solutions (involving both the IT system and possible organisational change) and their consequences at the same time as specifying the problem, thereby refining the understanding of the problem and developing the solution by an iterative process (Harker,Olphert, & Eason, 1990).

I have found it useful to represent the general requirements engineering process as four broad interactive component subprocesses, namely scoping, modelling, requirements elicitation and preference structuring. See Figure 4.3.1 and section 4.3 for a more detailed description of the process model. It should be noted that the processes and subprocesses themselves are in no way sequential. In addition, there is no set route through the process model as the direction of the process itself is governed by interactions between the problem solvers and problem owners.

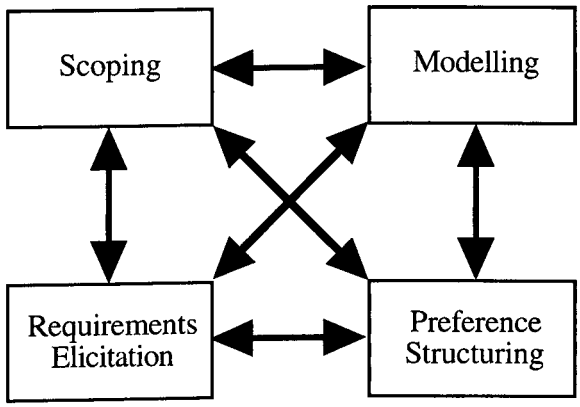


Figure 4.2.1.1 An Outline of the Process Model

For example, in contrast to other methods, modelling may be started at a very early stage to help in exploring the system boundaries and in identifying stakeholders. This process is in contrast to the more traditional 'waterfall' approach to modelling in which the output from one stage forms the input to the next stage and so on, with all the stages following a pre-determined order. The objective of the process of modelling is to support the identification and transformation of organisational requirements into precise statements. These statements can be operated upon by systems designers without being prescriptive as to the order in which the various operations involved in this process are carried out.

4.2.2 The Method

In hard systems thinking the computer is viewed as a pure computational object that manipulates data in some fashion so as to produce more output. However within socio-technical systems thinking computers are viewed as the medium through which people interact. As a direct result of this a better understanding of the organisational, informational and technical values that the system is required to encapsulate is defined. The use of this style of thinking leads us to examine and analyse the nature of the relationships that may exist within the organisational system and leads us to a more sociologically aware method. As a direct result, issues and questions such as what skills are required in order to interact with the technology and what manner should that interaction take can be identified and their answers examined and analysed.

The three main components of the requirements engineering method that is used in this thesis are defined as follows :

- Interaction and Iteration with the Problem Owners and Problem Solvers. (Interaction and Iteration - Section 4.2.3)

- Four concurrent sub processes each interacting with and supplementing the other. (The Process Model - Section 4.3)
- Five projections with their associated architectures and models (Projections - Section 4.2.5, Chapter 1, Chapter 3 and Chapter 5)

The question of how the three components of the requirements engineering method interact and relate to each other needs to be addressed.

4.2.3 Interaction and Iteration

In certain methodologies such as JSD, interaction between the problem owners and the problem solvers only occurs when a problem owner is giving a statement about the nature of the perceived problem, or when a problem solver is representing a statement about the nature of the perceived problem and its implications back to the problem owners. Interaction with the problem owners is limited and when it occurs it is very controlled, structured and managed.

Within the method presented in this thesis, interaction and iteration with the problem owners is seen as not only necessary but as essential to the success of the method. As was stated in Chapter 1 there are two types of uncertainty that need to be addressed. The first type is that of problem engineering uncertainty, i.e. can we solve the problem that the problem owners have? The second type is that of problem nature uncertainty, i.e. have we defined the correct problem?

Interaction and iteration at all stages in the software process is one way of addressing the problems of problem engineering uncertainty and problem nature uncertainty. The first type of uncertainty is eliminated as the interaction and iteration allows for the examination, classification and verification of what is possible and feasible. The second type of uncertainty is reduced as the interaction and iteration functions in such a way as to build consensus and thus more nearly achieve the utopian state of complete system specification and comprehension.

4.2.4 A Process Model

The components of the process model relate to the interaction and iteration between the problem owners and problem solvers by providing a fixed medium through which the process of interaction and iteration may take place. It is this process that builds consensus and eliminates the various types of problem uncertainty.

This is a model of the processes of scoping, requirements elicitation, modelling and preference structuring. One of the main characteristics of the process model is the way it has separated these four functions and shown the relation between them. Requirements are not considered as butterflies: they are not there to be captured and pinned down in a specification cabinet. Rather, the process of finding them is one involving at least three roles (requirements owner, requirements elicitor, requirements modeller) and a number of separate tasks involving some interesting feedback loops. The process model that will be used to set the context within which the enterprise modelling language and information modelling language operates is presented in Section 4.3.

4.2.5 The Projections

In (ANSA, 1990) five modelling projections or viewpoints are defined. However for the purpose of this thesis only the enterprise and the information projections will be further defined. See Chapter 1 for a description of each of the five projections. See Figure 4.2.5.1 for a description of the structure of a projection.

Each projection defines a set of issues and concerns (See Chapter 1). In representing and attempting to answer these issues and concerns each projection makes use of an architecture. The architecture is used to construct and validate a set of projection models, where the projection models function to address the issues and concerns of the projection (See Chapter 3). A projection model is constructed, analysed and validated through the use of a modelling language. See Chapter 5 for a detailed description of the Enterprise and Information projection modelling languages. The architecture itself consists of two components: semiotic and hermeneutic. The hermeneutic component is comprised of a set of axiological concepts. Axiology is defined to mean the theory of values and, in particular, a study of the ways in which values provide a basis for judging and deciding on actions. The enterprise projection makes use of the axiological concepts of responsibility, obligations and role. The information projection makes use of the axiological concepts of contracts, conversations and role. The semiotic component is comprised of a set of syntactic concepts. Syntactic concepts are concerned with the process model, rules, guidelines and objects that govern the use of a language. The conceptual framework is used a) to capture axiological concepts and embed them in the modelling language, and b) to map between the various projections. See Chapter 3 for a description of the enterprise and information projection conceptual frameworks. In order to capture and embed axiological concepts in the modelling language the conceptual framework contains both axiological and syntactic concepts. The concepts which the conceptual framework

makes use of are part of both the semiotic and hermeneutic components of the architecture. The modelling language addresses the set of issues and concerns defined by the projection by embodying axiological syntactic concepts.

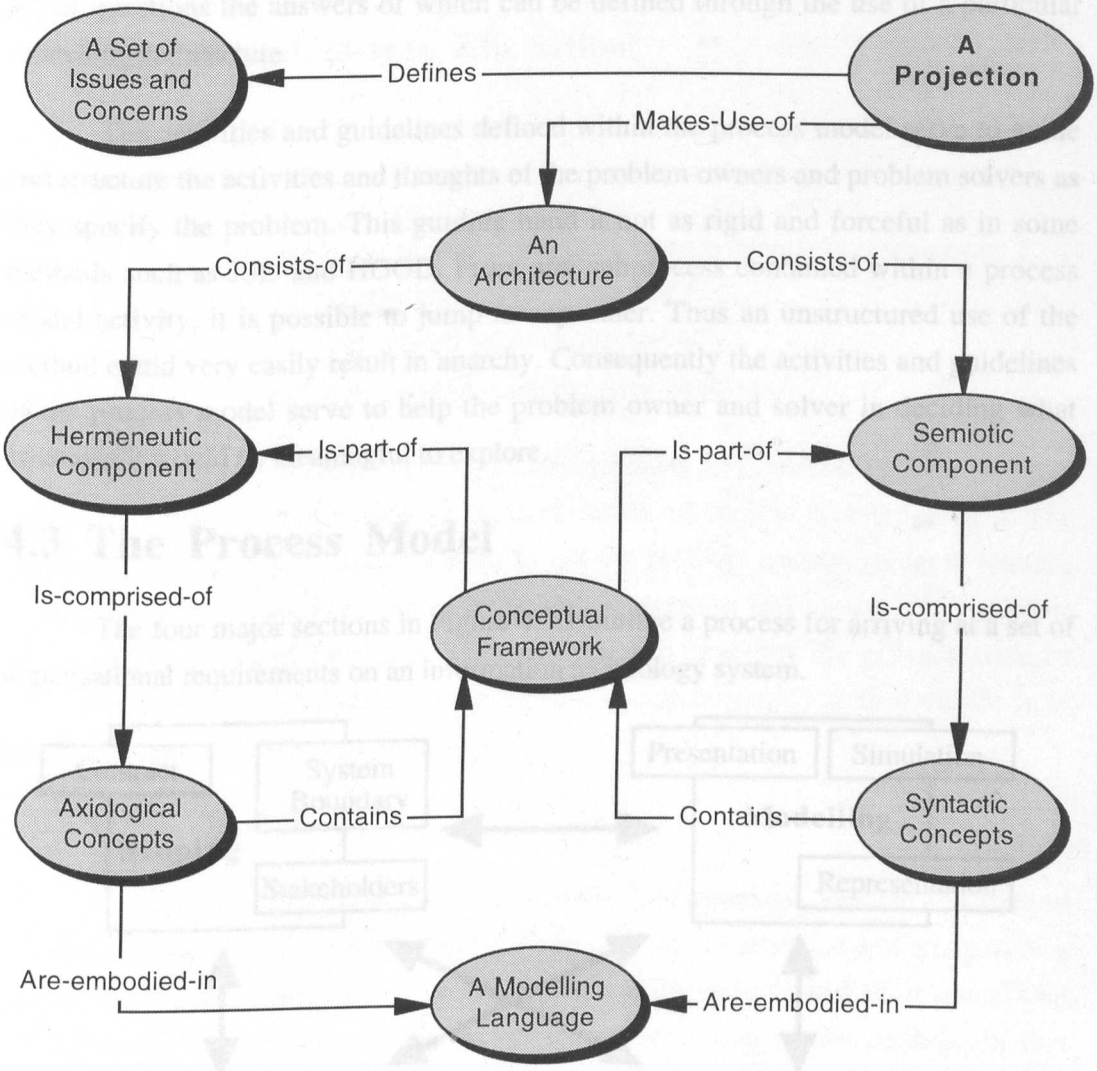


Figure 4.2.5.1 Components of a Projection

4.2.6 The Relationships Between the Components of the Requirements Engineering Method

The projections provide a conceptual modelling language and framework from within which it is possible to represent and analyse various organisational issues. These languages function much the same as any other language; their purpose is to act as a medium through which information can be exchanged and a consensus built. Each language stresses a particular set of issues and attempts to elucidate and represent a particular set of requirements. Consequently each language has a set of unique constructs and a set of constructs which it shares with the other projection languages.

The process model defines a set of activities which utilise the architectures defined by the modelling languages. These activities guide the problem owners and problem solvers through the complex maze of systems analysis. Each activity defines a set of questions the answers of which can be defined through the use of a particular modelling architecture.

The activities and guidelines defined within the process model serve to guide and structure the activities and thoughts of the problem owners and problem solvers as they specify the problem. This guiding hand it not as rigid and forceful as in some methods such as JSD and HOOD. From any subprocess contained within a process model activity, it is possible to jump to any other. Thus an unstructured use of the method could very easily result in anarchy. Consequently the activities and guidelines of the process model serve to help the problem owner and solver in deciding what questions it would be meaningful to explore.

4.3 The Process Model

The four major sections in Figure 4.3.1 outline a process for arriving at a set of organisational requirements on an information technology system.

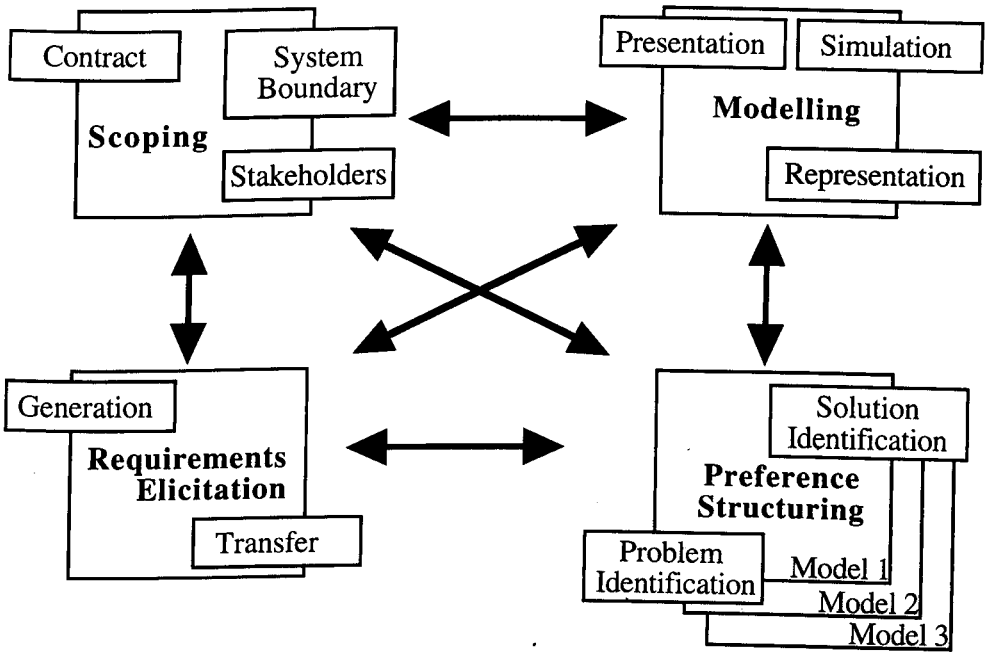


Figure 4.3.1 A Representation of the Method.

4.3.1 Scoping

This component subprocess deals with determining the scope of the requirements determination team and the contractual agreement with the client,

establishing boundaries for the contract, gaining an understanding of the purpose and structure of the organisational unit(s) which are to be involved, and identifying the principal stakeholders involved.

4.3.2 Requirements Elicitation

The primary function of the requirements elicitation process is to enable the problem solvers and problem owners to answer such questions as:

- What are the set of values that the system is intended to incorporate ?
- How will the requirements be evaluated ?
- Do we have an agreed set of requirements
- Are we communicating using the same language ?

Statements of the requirements are elicited from, and generated by, the problem owners. Statements of requirements are then transfered back to problem owners. The purpose of the process of transfer is a) to get the problem owners to agree that the correct requirement has been captured, and be) to exercise and consequently validate and establish a shared language. This process also involves the classification of requirements in order to identify conflicts, particularly conflicts in definitions of boundary objects and organisational aspects.

4.3.3 Modelling

The purpose of this component subprocess is to represent, validate and analyse our understanding of the socio-technical system by the construction and analysis of a set of models. This subprocess makes use of the projections and their associated architectures for the construction, analysis and validation of the models. In this subprocess we can also conduct the representation and formalisation of the problem in order to explore the nature and consequences of requirement statements, and to identify further areas in which hidden requirements may be revealed. This provides not only information about the environment in which the IT system is to function, but also a context for understanding later policy and design decisions. In this section we can also simulate our models of the system in order a) to validate our understanding of the problem, b) to draw out implications of the models of the system and c) to detect problems and contradictions that may exist within the system. Each model represents a stakeholder's particular view of the problem. Consequently this section also addresses the issue of traceability of requirements by allowing us to tag each model with the stakeholder to which that model belongs.

4.3.4 Preference Structuring

In this subprocess we take the models that have been constructed and rank these models according to the preferences of the stakeholders. This ordering over the models allows the problem owners to explore with the problem solvers the implications of a particular ordering. In addition, it allows us to identify conflicts among the problem owners and problem solvers.

In examining the ranking of the various models we can draw out the preference structuring upon which decisions concerning the problem and its solution will be made. Through this preference structuring of models we can identify the set of models that constitute a description of the problem and we can also make a decision about the set of models which constitute a description of the solution.

In this subprocess we can also identify and select conflicting or competing requirements. Preference structuring over models facilitates the traceability of requirements by allowing us to identify a) how decisions are made concerning the selection of requirements, and b) who makes those decisions.

Chapter 5 *Enterprise and Information Models*

5.1 Introduction

When developing a modelling language to model organisations there is a fundamental question about the nature of the models that needs to be resolved. This question is more about the philosophy and architecture behind the modelling language than it is about the model itself. In order to understand and use the modelling language to construct a model we must first comprehend what is meant by the term model, and the purpose to which that model will be put. The Concise Oxford Dictionary defines a model as "*a simplified description of a system to assist in calculations or predictions*". This definition is not far from the purpose to which a model of an enterprise will be put. The purpose to which an enterprise model will be put defines the functionality required of the enterprise projection, where the term projection is used to denote a particular viewpoint of an object or concept.

In (Zachman, 1987) an architecture is defined as a medium through which concepts may be expressed, examined and communicated. In Chapter 3 the enterprise and information projections are defined as a means of elucidating and expressing organisational requirements via an architecture. The primary purpose of both enterprise and information models in the enterprise and information projections, is to act as a bridge from the problem domain to the solution domain via some medium.

If the word "apple" was removed from the English language then we would be unable to refer directly to apples. A modelling language defines what we can talk about. The set of rules defines how we can glue words together from our modelling language to make statements. For example, in English we have the rule that every sentence must contain a verb. The rules for our modelling language tell us that the relationship between certain objects must be of certain types. The recipes and guidelines help us to

construct meaningful models and they represent a set of heuristics. So for example, when constructing enterprise models for the health domain, there will be recipes and guidelines specific to the health domain which will help us and point us in the correct direction. In constructing and analysing enterprise models for various domains we should always be attempting to identify recipes and guidelines that represent our success or failure. The result of all this identification is that we will build up a shared knowledge of what works and what doesn't, and those people who come after us will be able to draw upon this knowledge. A prefabrication is really a template that we fill in the blanks for when constructing a model. It represents some kind of generic knowledge about the problem domain. So for example, when constructing a model of a house we could use a template that says that a house has four walls, one door, two windows (front and back) and a roof. Then in ourepresents

5.5template and define what the roof, windows, etc. looked like.

The enterprise model defines a framework within which the flow of such things as responsibility and obligations can be expressed, discussed and examined. The aim of the enterprise projection is to facilitate the problem owners and problem solvers in comprehension and construction of the model. In the following sections what will be presented is the usage of a modelling language that allows decomposition that in turn allows for the modelling of any organisational structure. The information model defines a framework within which the flow, life cycle and meaning of information can be examined. Information is a social concept and this framework provides a context driven approach from within which axiological value and concepts can be explored and defined. Using concepts such as contracts and conversations the information projection allows for the flow and accessing of information to be defined and explored. The aim of the information projection is to facilitate the problem owners and problem solvers in comprehension and construction of the model.

5.2 The Basic Metaphor

> *"No object is an island. All objects stand in relationships to others on whom they rely for services and control"*
>
> - (Beck & Cunningham, 1989)

This statement may seem simple and obvious but it is fundamental to the notion of modelling information and information flow. Information is a concept that only has meaning when considered within the social contextual setting that forms part of its environment. For example, the number 8756 is meaningless unless you are told that it is a pin number for a credit card. In identifying this a whole set of relationships between the number and other objects are also identified. Consequently it is this set of

relationships that give the number its informational meaning. Organisational models need to be viewed as structured socio-linguistic phenomenon. As organisations can be viewed as societies of cooperating and interacting objects, any model that attempts to elicit requirements should encompass both the functional and social aspects that form the organisation.

I believe that any modelling technique that attempts to capture and represent an organisation should possess two very distinct aspects to its expressive power. These aspects should allow for the clear delineation and examination of the social and functional ways in which objects can relate and manipulate each other. Let me illustrate this with an example. Object *A* can have a functional relationship with object *B*, this relationship being of type "*Request more data*". While this relationship clearly captures the functional structure that exists between these two objects, it in no way captures the social nature of the relationship. Consequently the problem solver fails to understand the true nature of the way in which the two objects relate to each other. By adding a social relationship of type "*Consumer - Supplier*" between the two objects, the context within which the functional relationship functions can be clearly examined and hence better appreciated.

The question that now has to be addressed is one of how to model objects and object interaction so as to capture adequately the functional and social aspects that together define the organisation and its behaviour?

5.3 The Enterprise Model

5.3.1 The Concept of Role

> *The ruler rules, the minister ministers, the father fathers and the son sons*
>
> - Confucius

A role is defined to be those behaviours characteristic of one or more persons in context (Biddle, 1979). In this thesis I consider a role to be a configurarble generic concept and for the purpose of this thesis the decision has been made to describe an organisation as a set of related work roles for the following reasons.

- A role is both a descriptive and a normative concept that can be used to represent many different organisational realities from the structured to the unstructured.

- Treating roles as a basic building block makes it possible to move between organisational requirements and the requirements of individual users. (e.g. from the organisation's role in a project to the way these responsibilities devolve to the roles of members of the project team).

- A role defines behaviour and thereby system requirements.

- A role defines the relationships between role holders and the behaviour they expect of one another, which in turn defines many environmental requirements.

When enterprise modelling we distinguish between two major, and distinct, concepts of a role. A structural role, which is a relation between agencies or agents, and corresponds to the social aspect of role and functions to define the context of the behaviour. Examples of structural roles are supervisor–subordinate, supplier–customer, provider–consumer, and so on. This is in contrast to a functional role which is a relation between agencies or agents, and corresponds to the behavioural aspect of role. Examples of functional roles are executor (of an activity), owner, controller and so on. Hence our concept of role allows us to distinguish the following:

- Agencies and agents with associated obligations and responsibilities to other agencies and agents (Structural Role - See Section 5.3.2).

- Activities that interact through the utilisation of resources and are structured into actions and operations (Functional Role - See Section 5.3.3).

This distinction between functional and structural roles enables us to represent and analyse the relations between functional and structural concepts and to express the way in which they operate in real organisations. A marked advantage of our modelling technique is the way in which we can compose and decompose our diagrams for the purposes of ascertaining requirements at various levels of agency (individual, group or organisation). It is through using the features of enterprise diagrams in this way that we are able to specify the differences between relationships. Our use of the abstract term 'agency', for example, is deliberate so that we can discuss who or what corresponds to the agency.

Through the concept of role we can explore the issues of "to what does the role bind ?" It is sometimes the case that binding turns out to be a fundamental policy issue. For example, it might be agreed that there should be a beneficiary from the existence of a market (e.g. the market landlord). However in the case of a public utility supply

market, there might be a debate as to whether this beneficiary is to be the public state, or the private shareholder.

An important point that this serves to highlight is that differing requirements are expressed at differing levels within an organisation. At the organisation level the most important requirements may be to do with policy issues, while at the group level they may be concerned with being able to support collaborative work in an efficient manner, and at an individual level priority may be given to job satisfaction. We believe that enterprise modelling is capable of capturing conflicting requirements within an organisation's structure. In addition to representing these qualitatively different requirements at their appropriate levels, this representation can be used to discuss issues of conflict resolution, compatibility, desirability and so on.

5.3.2 The Concept of a Structural Role

5.3.2.1 What a Structural Role Represents

While agents act as the primary manipulators of the systems state, agencies act as repositories for responsibilities and obligations, and structural roles act as their binding points. A structural relationship serves as a means for the responsibilities and obligations to flow from one agency to another and thus responsibilities and obligations flow through an organisation. In Figure 5.3.2.1.1 the purpose of the structural and functional roles within an organisation are depicted.

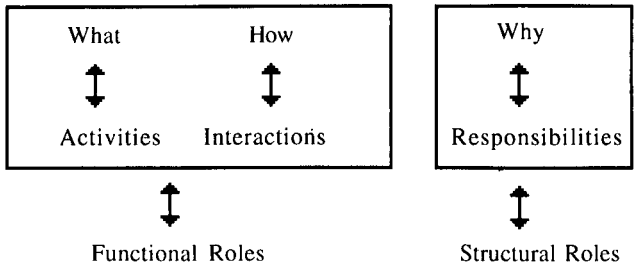


Figure 5.3.2.1.1 Role Diagram

A structural role is defined by the set of responsibilities that bind to it. Each responsibility in the set in turn defines a set of obligations. Each obligation in turn defines either a set of tasks that the role holder is engaged in performing or a state of affairs that the role holder is engaged in either maintaining or bringing about. These tasks and states of affairs are derived from the problem owner and should reflect the nature of the enterprise projection.

The key to understanding the nature of structural roles and their relationships with each other is in understanding the primary purpose of the enterprise projection and the uses to which it will be put. The enterprise projection facilitates a problem solver to model, and to comprehend, how organisational attributes like obligations and responsibility are established, flow through an organisation and are then discharged or fulfilled. By understanding such organisational attributes the problem solvers are better equipped to comprehend how the organisation interprets not only such attributes but also the binding of such attributes to agencies within the organisational setting. It is believed that by understanding how the organisation interprets such attributes the problem solver can better understand how a computer system would function within the organisation.

Structural relationships of the particular types and under a particular set of circumstances may be transitive in nature. That is to say that if the agency *A* has a structural relationship *S* with the agency *B*, and the agency *B* has a structural relationship *T* with the agency *C*, and given that the structural relationships *S* and *T* are of the same type and a certain set of conditions hold, then we may say that the agency A has a structural relationship with the agency *C* provided that the agencies *A* and *C* share the same parentage. A requirement on the notation is that it allow us to express and describe the types of relationships and circumstances under which they are transitive.

The set of structural roles that an agent can hold is divided into three types, a power relationship, a peer relationship and a service relationship. These relationships are described in the following sections.

The Peer Relationships - The peer relationship is a far more subtle relationship than the power relationship as this relationship appears to be more social in nature than the power relationship. In a peer relationship two or more agents share a common power relationship with a third agent. It is important to note however that this power relationship should be of the same type. In a peer relationship there is no implication of enforcement, in fact, it is exactly the lack of this attribute that is characteristic of peer relationships and makes them special. Consequently when two agents are in this relationship they may request that each other perform various services, but they lack the facility or the power to enforce execution. As a result agreement to perform a service is achieved by means of negotiation. An example of a peer relationship is that of the colleague relationship.

The Service Relationships - In a service relationship one or both of the agents have the power to invoke the execution of a predefined and agreed task by another agent. This task will in some way relate to both the invoking and executing

agents. An example of a service relationship is the consumer—supplier relationship, an example of which is the relationship that most people can be said to hold with an electricity board. In this relationship one agent acts as the consumer of a service while another agent acts as the supplier of that service. The difference between a service relationship and a power relationship is that when the consuming agent is dissatisfied with the service provided by the supplying agent then the consuming agent may appeal to a third agent. It is this third agent that has the ability to enforce its judgements on both the supplying and consuming agents. A service relationship is in essence one agent invoking the performance of a predefined activity by another agent with predefined rules for the enforcement of the correct execution of that task.

The Power Relationships - The essence of a power relationship is that one agency has the power to make and enforce demands on another agency. It is important to note however that the enforcement of these demands may be made via a third agency. An example of a power relationship is the supervisor—subordinate relationship that can exist in most organisations. There are however many different types of this relationship, for example master—slave. In this relationship the supervisor has the power to define the responsibilities and obligations that a subordinate is required to fulfill, and to judge whether or not the responsibilities were correctly discharged. The subordinate is not totally subservient to the supervisor in that the responsibilities and obligations that the subordinate is required to fulfill are defined by means of interaction between the two agencies. The types of power relationships that can exist between two agencies within an organisation can be defined with reference to the types of interactions that are meaningful for the two agencies to engage in. For example if a person's boss punishes them and they think that the punishment was unfair then they may appeal to a higher authority, the final authority being, of course, the law courts. The nature of these relationships is very complex and something that the problem solver would need to explore carefully and in depth with the problem owners.

5.3.2.2 Explications of Structural Relationships

The structural relationship diagrams that will be introduced in this section are normative. That is they attempt to explain what is required for a particular structural relationship in order for it to be such a relationship. Therefore we term such a diagram an explication of the structural relationship that it represents. The modelling language which is used to construct and analyse explications is presented in section 5.3.4.

When modelling the structural relationships that can exist within an enterprise there are two questions that the model should be able to answer. The first question is w"hat responsibilities and obligations are allowed to exist within the relationship?" The

second question is what "enterprise objects, i.e. resources etc., must exist in order to support and give meaning to the responsibilities and obligations?" It is important to note that these objects can act in several ways. Firstly they can act to show that a responsibility or obligation has either been established, discharged or altered in some way. Secondly they can act as objects over which responsibilities and obligations are established. Finally they can act as a medium of communication.

For example, a problem specification document may act as a token of an agent's acceptance to perform an action and thereby the creation of a responsibility. The third and final question is "under what conditions are the relationships between the objects valid?" E.g. an identifier and password for a computer are only valid between the hours of 8:00 am and 9:00 pm, or, a ship's captain can only perform the marriage ceremony while the ship is at sea. The answers to these questions can be examined by placing the access to the object, and the creation of the attributes of the object within the context of an interaction. Part of the purpose of a structural relationship is to define what types of interactions the two agencies may engage in. Consequently implications of changes to objects and attributes of the system may be explored and examined.

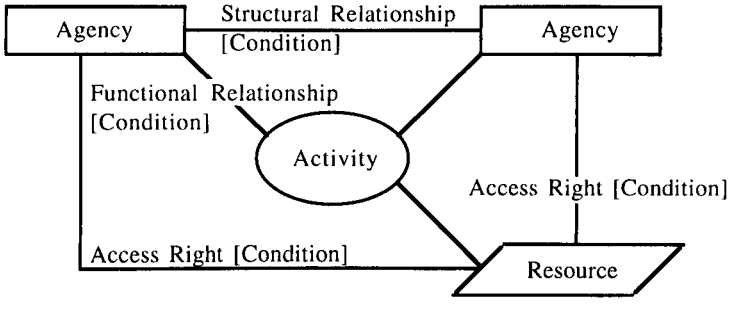


Figure 5.3.2.2.1 Structural Relationship Diagram.

A structural relationship diagram is depicted in Figure 5.3.2.2.1 and in this diagram there are a few things that should be pointed out. The first is that each object type is represented as a distinct shape. i.e. the agents are drawn as rectangles, the tasks as ovals and the resources as rhomboids. The type of arcs i.e. access rights are defined by the enterprise icon (see Chapter 4). The arcs also have a condition associated with them. The task that is shown at the centre of Figure 5.4.3.1 is derived from the responsibilities and obligations that a particular agency may hold. Responsibility is a three place relationship between two agencies and a state of affairs. For this relationship we say that the agency *A* is responsible (in some way) to the agency *B* for bringing about or maintaining a state of affairs. It is from this that the task definition is derived. A structural relationship diagram can be used in one of two ways by the problem owners. The first is to help them in their task of requirement elicitation by prompting

them to ask certain questions. For example "when and under what conditions is this relationship between two objects meaningful?" The second is in allowing them to explore the ramifications, implications and possible contradictions of policy statements.

5.3.3 Functional Roles

5.3.3.1 What a Functional Role Represents

Functional relationships link together two functional roles in different agents or agencies where each agent or agency is called a role holder. One of the purposes of functional relationships is to define the behaviour that a role holder may engage in with another role holder within the context of a structural relationship. We may say that one of the purposes of a structural relationship is to define the context for a functional relationship. In defining and modelling the behaviour of a role holder, the problem solvers are in fact defining and modelling the set of allowable functional relationships that can exist for that particular role holder.

Functional relationships aid in the identification of the organisational objects that are required to give meaning to the interaction. The purpose of a functional relationship from the perspective of its role holders is to facilitate the correct discharge of their responsibilities and obligations. The behaviour that one role holder may engage in with another takes the form of interactions. The context of these interactions is defined by the functional relationship within which they are said to take place.

From the perspective of functional relationships organisational objects can be classified into one of two types. The first type of object is the enterprise attribute of either obligation or responsibility that are placed upon role holders. These enterprise attributes are placed upon the role holders by virtue of either the tasks that they are required to perform, or the state of affairs that they are required to achieve or maintain. The second type of object is the enterprise object and its relationships that are required to be present in order for the role holders to discharge successfully their enterprise attributes. The modelling and defining of the life cycle of both enterprise objects and enterprise attributes is achieved via the modelling of the set of interactions that the role holders of an organisation may engage in.

5.3.3.2 Interaction between Role Holders

In the enterprise projection the interaction between two role holders defines how, when, where and under what circumstances projection attributes like obligations and responsibilities are established, flow through the organisation and are finally

discharged or fulfilled. By modelling the life cycle of such attributes the problem solvers may attempt to answer a number of types of questions.

The first type of question allows for the examination of the possible conflicts that could arise for any given role holder. The term conflict is used to denote a situation where a role holder is either obliged or responsible to perform an action, or bring about some state of affairs, whilst at the same time being obliged or responsible either not to perform the action, or not to bring about some state of affairs. The second type of question is concerned with the elucidation of the conditions under which an agent can either not discharge an obligation or not fulfill a responsibility. The third type of question is concerned with the elucidation of what objects act as tokens of either obligations or responsibilities. The fourth type of question is concerned with the delineation of the valid accesses to objects that act as token of either responsibilities or obligations. The fifth and final type of question is concerned with the examination and comprehension of the correct creation and deletion of the objects that act as tokens of either responsibilities or obligations.

5.3.4 The Basic Components of a Modelling Language

In order to represent the complexity of a social model of an organisation there is a need for a basic ontology. The purpose of the ontology is to define a language and grammar from which an organisational model can be constructed. The basic ontology is defined to consist of agents, activities and resources.

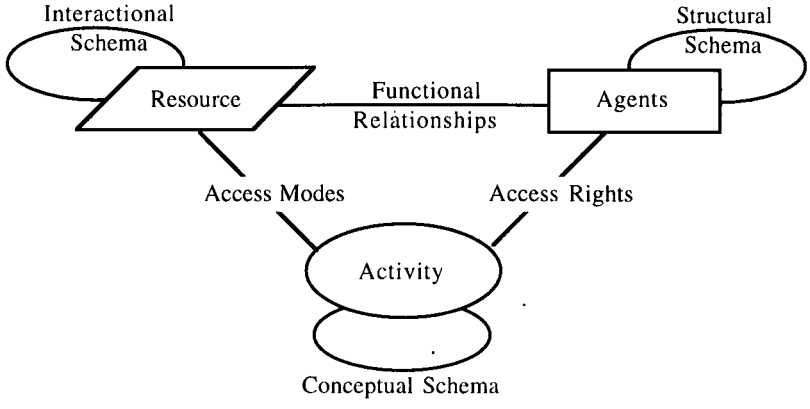


Figure 5.3.4.1 Basic Entities and Relations

This ontology is taken from the activity level of the enterprise conceptual framework depicted in Figure 3.4.2.1 and is presented in Figure 5.3.4.1. Every concept that was presented in the conceptual framework depicted in Figure 3.4.2.1 is present in the modelling language, and is constructed from the basic ontology presented

in this section. For example, a responsibility is a particular type of relationship that can exist between only two agents.

- **Agent**

Agent is an extensional concept, in the sense that an agent is the physical manifestation of the holder of a set of responsibilities. An agent may be regarded as the primary manipulator of the state or structure of the system. It is the only object that through actions can create, modify or destroy other objects. This may be done only by virtue of the responsibilities associated with the agency that the agent holds. Agents should exhibit the following properties (Woolridge & Jennings, 1995):

- Autonomy : agents have some kind of control over their actions
- Social ability : agents interact with other agents via some kind of agent-communication language and communication medium.
- Reactivity : agents perceive their environment and respond to changes that occur in it.
- Pro-activeness : agents do not simply act in response to their environment, they are able to exhibit goal directed behavior by taking the initiative.

A agent must always be an appropriate answer to a "Who is responsible for . ?" question. An agent is not necessarily a human; an agent can be human, mechanical or some combination of the two, as for example when a computer acts on behalf of some individual.

- **Activities**

Activities are the state changing operations that can be performed on the system by agents. An activity must induce a change visible by some agency, though not necessarily at the time it occurs. Everything that happens in the system must be in principle traceable to some action, though not all actions are intended, nor do intended actions always behave as expected.

- **Resources**

Resources can be one of two types, physical or logical, where physical resources are tangible objects such as nuts and bolts, and logical resources include information and time. When modelling organisations at the enterprise projection level, resources act either as tokens of responsibility signifying that an agent has a binding responsibility upon it, or as objects for which some agent is responsible. An important

type of logical resource is data. When data is passed from one action to another interaction is said to occur, the data being the bearer of those interactions. Data is the bearer of interaction between actions. That is one action creates data that another action uses, possibly non–destructively. Of course this may generate certain constraints that may be specified in terms of temporal ordering, the necessity to share resources and so on; but all these are secondary aspects of the interactions. What is important is that one action interacts with another through the exchange of data. There is no data that is independent of interaction.

- **Agent - Agent Relation**

We have already indicated in general terms that the nature of the relationship between agents is that of responsibilities and obligations.

- **Activity - Activity Relation**

Essentially there is only one type of relation that can exist between activities: that of interaction. Interactions have two characteristics of interest: the resources that mediate the interaction, and their temporal (partial) orderings. A logic of interactions will have to be capable of expressing these aspects and many such logics already exist (Galton, 1987).

- **Resource - Resource Relationships**

We have defined data as the bearer of interactions between actions. Now we need to mention the relationships that can exist between any resource, physical or logical, that are independent of any actions. These relationships form what in data modelling terms is called the conceptual schema. There is a variety of forms it may take, and the choice may in any instance be dictated by convenience or policy. Examples might be a relational model of data in one of its many forms, a hierarchical model or a network model. Remember, however, that resources can be physical as well as abstract. We now turn to a discussion of the kinds of relationships between the three basic entities in our model.

- **Agency - Activity Relation**

If we examine the kinds of relationships that can exist between agents and activity, we find that they are all expressed in social terms: Observer, Executor, Customer and so on. For example, we can make some elementary distinctions between the relationships as follows:

The **Observer** of an action knows that it is taking place and may, or may not, know of any of the relationships which now follow.

The **Owner** of an action has the right to destroy it; (the owner of an action may differ from the creator of an action, since ownership can be transferred).

The **Customer** of an action has the right to change its specification.

By the *right* we mean two related things: a capability exists to perform the action and this capability by virtue of some legal instrument can be enforced by recourse to something outside the system (e.g. judicial).

- **Action - Resources Relations**

We can characterise the relations between actions and resources as being of various **access modes**: for example reading, writing, allocating, consuming and so on. It is important to note that although an access mode is a relationship between the action and a resource, there may well be simultaneous policy constraints on both the relations between the actions and the resource, and the relations between the accessing action and some agency. For example, certain tools might be permitted access only when wielded by certain agencies.

- **Agent - Resource Relations**

We have already introduced *rights* in terms of the functional relationships. We now want to refine this notion. An agent may derive certain rights over an activity by virtue of the relationship between agent and activity. One important class of such rights is those that exist in respect of the resource that the action accesses. That the rights over the resource are not necessarily dependant on the rights over the action can be seen from the following example. A manager may well choose to terminate some action in the organisation, but this does not necessarily give him the right to destroy the resources that the action may have generated.

Again, the executor of an activity will not in general have the right to destroy the activity. Such resources may still be required for auditing or historical purposes. Again, the executor of an action will not in general have the right to destroy some transient, relatively valueless resources produced by it—temporary working notes, administrative memoranda and so on. In computational terms the distinction may be between stack frames that are transient data, and log files containing audit data. In general then, we can distinguish between rights over an activity from rights over resources by modelling the latter as a direct relationship between agents and resources.

5.3.5 Modelling Enterprises - An Architectural Approach

In this section the aims and objectives of a particular organisational representation will be presented and discussed along with the organisational representation itself. A representation of an organisation can be constructed from a set of templates where the term template is used to denote an enterprise architecture prefabrication.

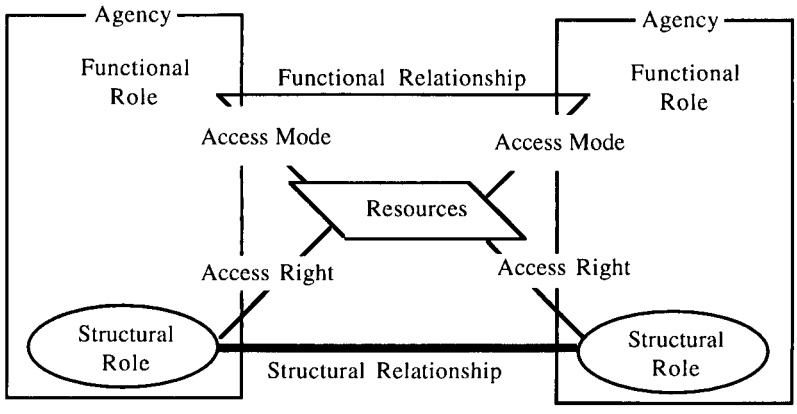


Figure 5.3.5.1 The Model Template

The primary purpose of the template, (enterprise architecture prefabrication), as show in Figure 5.3.5.1 is twofold. The first is to provide a language template from which a set of enterprise diagrams can be constructed which represent a particular view, perspective, or perception of the system as defined by the enterprise projection. The second is to aid in the construction of explication of structural and functional relationships by focusing the problem solvers on the resource objects that exist for a particular agency.

The key to understanding and using the template is to understand the nature of the resource that lies at its centre. This resource can act in one of two ways. The first way in which a resource can act, is as a token of either an obligation or responsibility upon an agent or agency. An example of a resource acting as a token of a responsibility is a service agreement that a university can have with the suppliers of its computers. In this example the resource acts in such a way as to define the services that university may request from the supplier. The second way in which a resource can act, is as an object which is required by the agent or agents holding a responsibility in order for them to fulfill their responsibility. An example of a resource acting in this way is money which exchanges hands when you purchase as house. In this example the money is acting as a resource which allow you to fulfill your responsibility to purchase a house.

5.3.6 Representing Organisations - An Enterprise Perspective

A set of enterprise diagrams can be used to model an organisation and thus to express and explore organisational issues and boundaries. An enterprise diagram shows the structural roles and functional roles, and the relationships that exist between them. It can therefore be used both as an analytical descriptive diagram, in that it attempts to show the relationships that actually exist within an organisation, and a prescriptive diagram, in that it can be used to represent the required organisational structure.

A detailed description of the problem that is used in this section to illustrate the expressive powers of a enterprise diagram is given in Chapter 7 along with a more detailed set of diagrams. However in order to appreciate adequately the true expressive powers of this notation a brief summary of the problem statement is required. The electricity board has a legal contract to service all electrical appliances that they sell for a fixed period of time, normally three years. The consumer performs two roles from the perspective of the electricity board when an appliance breaks down, that of owning the problem and that of reporting the problem. The electricity board (supplier) performs two roles from its perspective when an appliance breaks down, that of problem report reception and problem repair. The electricity board internally consists of three distinct agencies, that of management, problem reception and problem repair. The management is legally responsible for supervising the behaviour of the other two agencies. The problem reception and problem repair agencies are related as colleagues and information about the received problem flows from the problem reception and problem repair agency.

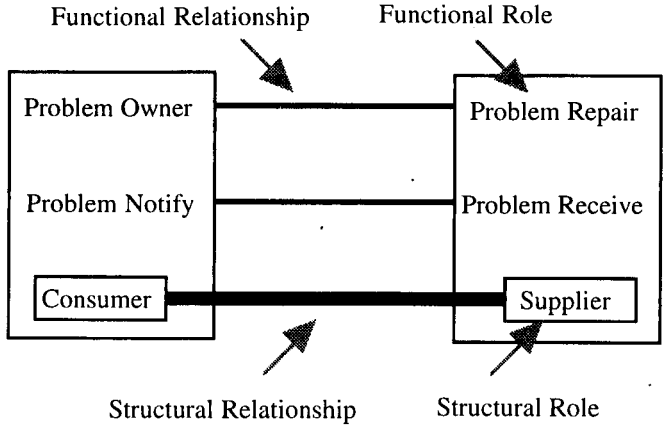


Figure 5.3.6.1 A Simple Enterprise Diagram.

In Figure 5.3.6.1 the resources have been removed to leave a clean representation of the organisation in terms of its basic entities and their relationships.

The term agent entity is used to denote either an agency or an agent, where the definitions of agency and agent are given earlier in this chapter. The first point to note about Figure 5.3.6.1 is that the rectangular boxes represent agent entities. Links between functional roles, shown by lines of single thickness, are called functional relationships, and links between structural roles, shown by lines of double thickness, are called structural relationships. Each box contains the names of some functional roles and some structural roles. The functional roles (*Problem Owner, Problem Repair, Problem Notify, Problem Received*) and their functional relationships are a shorthand for the kinds of behaviour that agent entities may engage in, and are used to represent interaction. It is through this interaction that obligations are transferred and discharged and by discharging its obligations an agency may fulfill its responsibilities. The structural roles and structural relationships are a shorthand for the framework of responsibilities that permit and give meaning to these behaviours.

Each agent entity depicted in Figure 5.3.6.1 can be decomposed into sub agent entities. When decomposing an agent entity we also decompose the roles associated with it. For example the supplier agent entity can be decomposed into a problem reception agent entity and a problem repair agent entity as depicted in Figure 5.3.6.2.

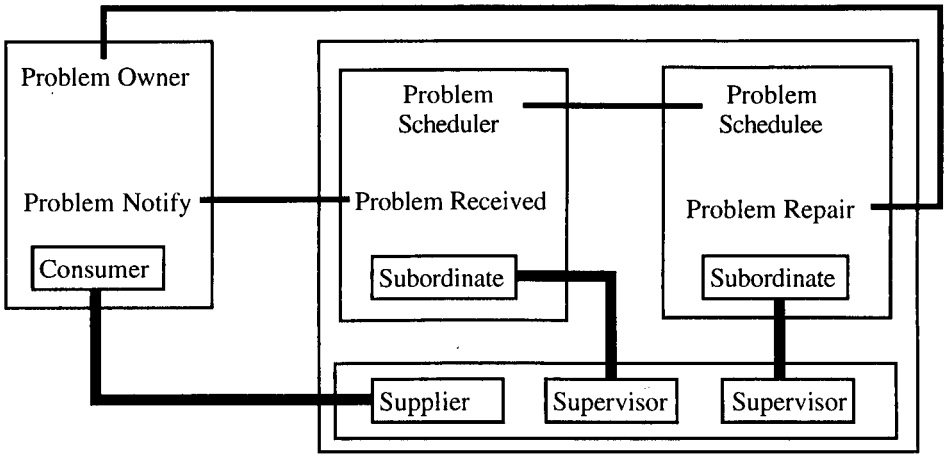


Figure 5.3.6.2 Decomposition of a Enterprise Diagram.

From the point of view of the problem reception agent entity the supplier agent entity from which it has been decomposed is called its parent, and the problem repair agent entity is called its sibling. This idea can be extended to grandparents and grandchildren of agent entities, i.e. any sub-agent of an agent entity views that agent as its parent, and that agent views its sub-agent as its child. The concept of parentage is one that can be applied not only to agent entities but also to role and relationships. Thus we may talk about and examine how a role within an organisation is decomposed and assign to the various sibling agent entities.

Each agent entity has a parent agent entity, and the parent agent entity of the organisation is called *world*. Consequently the structure that is formed by the decomposition of the organisation into its sub components can be viewed as a tree, where the root node is called the world. The *world* has some special properties that should be noted. It contains no structural or functional roles and has no obligation relations or instrumental interfaces linking into it. It is an agent entity in name only. It does however allow the structure of the decomposition of agent entities to be viewed as a tree.

When constructing an enterprise diagram there is an obvious question that needs to be addressed, and that question is where to start and stop in the construction process. The answer to this question is not as simple as it might at first seem. In fact the answer to this question is one of personal choice and it is to answer this question that the problem solvers would draw upon their knowledge about the problem domain that they are dealing with.

5.4 The Information Model

5.4.1 Introduction

The currently existing models of information and information processing such as (Chen, 1976; Devlin, 1991; Mills,Linger, & Hevner, 1986) all draw upon the idea of state and state transitions in some form or another. They all fulfill the requirements of a data engineering model, in addition to some of the requirements of an information engineering model. They all however fail to address the information engineering question of "what is the nature of information ?". The reason for this is that information is far more than just data plus a set of formal methods for manipulating that data. I believe that the true nature of information can only be examined when one examines the social system within which it is set.

Information processing within organisations is concerned with bargaining, supporting the fulfillment of contracts and decision making. In this view, the main job for an information technology system is to facilitate co-ordination between separate enterprises or groups within an enterprise, each of which possess their own policies, objectives, values and interests. It is for this reason that we model information from the contractual perspective rather than from the current structure of the information flow. An IT system can be seen as a medium of communication, serving as an intermediary in a social process. The model of information which will be advocated in this thesis is one that uses conversations and contracts to describe the context of application of a particular item of information. The type of conversations that can exist within an

organisation are derived from the responsibilities, obligations, and role-relations expressed in the enterprise model. For instance, we could distinguish between a request or a command to perform a particular service, the implications of both being structurally different.

5.4.2 A Contractual View of Information

In order to understand correctly the organisational information system it is necessary to comprehend fully the social systems that form its environment. The concept of information can be viewed as social, and information can be defined to only exist within a social system. From the perspective of the organisation and the information projection, information is only meaningful when considered along with the contractual relationships and conversations that define it. Thus a four digit number is meaningless unless the contracts and the conversations that define the relationship between the owner of the four digit number and the bank are also defined.

There is a substantial body of law that exists in the legal system relating to the notion of a contract. This idea for a formalised agreement between any number of parties is one that can be traced back many centuries and through many cultures. In fact much of the legal accumulation today is expressly to deal with what precisely constitutes a contract.

The idea of using the concept of a contract as a conceptual means of elucidating problems first appeared in the area of artificial intelligence. In (Hewitt,Bishop, & Steiger, 1973) the concepts of contracts and actor were first introduced where a contract was the interface that one actor made available to another. Programming languages such as PLASMA (Hewitt, 1977) and ACT1 (Leiberman, 1987) were then developed. In additional (Davis, 1983) draws upon the idea of contracts in the design of a *contract net.* In this a contract is viewed as a network of entities where they can make requests and negotiate contracts for the performance of some service.

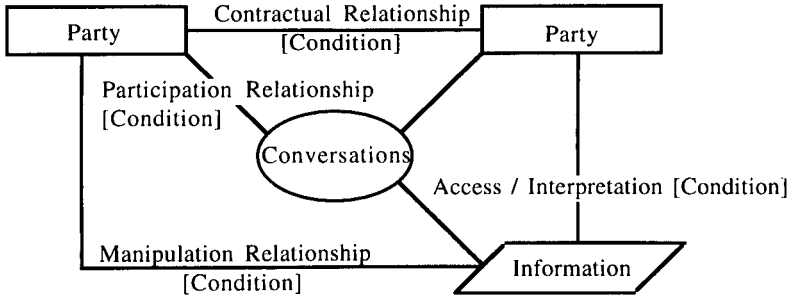
5.4.3 Modelling Contracts

In every type of contractual relationship there are two types of contractors called role holders. Each role holder performs some function with reference to the contractual relationship. From the perspective of the information projection a contractual relationship can define a set of four other basic components, each of which has its own specific function and purpose to fulfill from the perspective of the organisational information system. The four components of a contract are now listed below.

- Contractual Roles and Relationships
- Participation in a set of Conversations
- A Set of Manipulations of Information Structures.
- Limitations

Representing the four basic components that form a contractual relationship yields a diagram akin to that of an explication of a structural relationship that was depicted in Section 5.4.2.

Figure 5.4.3.1 shows the pictorial representation of all four of the basic components that form a contract, and how they relate to each other. This diagram is used at a high level to elucidate the semantics, consistency and completeness of an organisational informational model.



5.4.3.1 A Contractual Relationship Diagram

5.4.3.1 Contractual Roles and Relationships

The contractual roles represent the nature of the service that is required by the contract to flow from one role holder to the next. The contractual relationships, and consequently the contractual roles that a party can hold, define the functionality that the party can possess from the perspective of the information projection. The nature of a contractual relationship is used to define the way in which the particular holders of the contractual roles may engage and participate in a conversation.

5.4.3.2 Participation in a Conversation

The way in which a contractual role holder may participate in a conversation is governed by the nature of the contractual relationship that the role holder is actively engaged in. The contract may be said to define a conversation as well as the way in which the role holders may participate in it. For example, if a homemaker is engaged in a client server contractual relationship with the electricity board the nature of this service is one of compliance. The homemaker may then initiate a conversation the nature of which is to demand that a particular service be rendered.

5.4.3.3 Manipulating the Information Stores and Information Structures

The primary purpose of a party's information store is to act as the repository for its information structures. Each party has their own information store, and it is the conversations that a particular party is engaged in that manipulate the information structures contained in the information store. The easiest way to illustrate the way in which conversations govern the manipulation of the information stores and information structures is with an example.

The user of a computer system can be said to have a consumer–supplier contractual relationship with that system. The process of logging onto the computer system can be viewed as a conversation between the user and the computer. So when the user tries to log onto the computer system the computer will request that the user enters their identifier. In supplying the identifier the user has to access their information store in order to retrieve the pertinent piece of information. Once this has been supplied and recorded by the computer, it will then prompt the user for a password. Again in supplying the password the user has to retrieve the relevant piece of information and the computer has to record it. Once the computer has both the identifier and the password it proceeds to access its information structures in order to authenticate the user. Thus both the user and the computer must have information representations of an identifier and a password. These representations may however be very different as they serve very different purposes.

5.4.3.4 Limitations

From the perspective of the legal system it is the limitations that a contract defines that are the most interesting. A contractual limitation may work in one of three ways. It may work as a precondition that must be satisfied in order for the contractual relationship to be a valid and meaningful one. Or it may function as a post condition that

must be met in order for the contractual relationship to have been correctly terminated. Finally a contractual limitation may serve as an invariant for the correct execution of the contract. It should be pointed out that the limitations can be of any form, shape or substance. That is to say that the limitation may be dependant on time or on the presence of another object. An example of a limitation is that an offer to purchase a house is valid for only 48 hours or the captain of a ship can perform the marriage ceremony only whilst the ship is at sea.

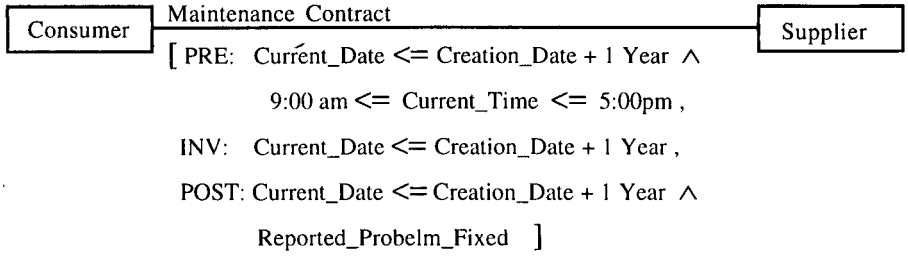


Figure 5.4.3.4.1 Limitations on a Relationship

An example of how a set of limitations governing the meaningfulness of a contractual relationship can be depicted is illustrated in Figure 5.4.3.4.1. In this example the type of contractual relationship is one of consumer–supplier. The supplier undertakes to provide some service at the request of the consumer and the nature of this service is one of maintenance. The pre, post and invariant conditions specify the conditions that are required to be satisfied in order for the relationship to be meaningful. The pre condition specifies that the maintenance relationship is valid for one year from its creation and that any service invoked as a result of this relationship must be invoked between the hours of 9:00 and 17:00. The invariant stipulates that while any service is being rendered the condition that the current time is within a one year period from the creation of the contractual relationship must hold. The post condition besides specifying the time constraints on the validity of the contractual relationship, also specifies the constraint that the service rendered by the supplier must satisfy the consumer.

5.4.3.5 What a Contractual Role Represents

The types of questions that the information projection attempts to address are primarily concerned with the elicitation, representation, formalisation and validation of organisational information requirements. The way in which the information projection attempts to capture information requirements is by modelling the contractual relationships that can exist within an organisation along with their organisational context.

In modelling the contractual aspects of the organisation the information projection attempts to capture, represent and validate organisational requirements concerned with the processing of, and access to, information. In order to capture correctly these requirements a classification of contractual relationships is essential. The primary objective of the classifications is to facilitate the answering of questions such as, "What are the types of contracts that can exist within an organisational setting and what are their ramifications?" The structure of the classification that is presented in the following sections attempts to answer these questions by drawing upon the rights and authorizations involved in invoking a conversation.

An implicit component of any contractual relationship is the monitoring and enforcement of the relationship by a third party. For the purpose of the information projection the role of monitoring and enforcement is performed by the legal system. The enforcement and the decisions rendered by the third party are all performed with reference to a set of formal rules. In any normal society such a set of rules and the process of decision making and arbitration by a third party would be called a legal system.

Consumer–Supplier - The nature of the consumer–supplier contractual relationship is one of a supplier delivering a predetermined service at the request of a consumer. The nature of the predetermined service that the supplier is required to deliver is something that is established via a negotiation process. This process is something that both the supplier and the consumer are required to have been engaged in prior to the service invocation. The key to understanding this relationship is in understanding that the service is requested by the consumer. The easiest way to illustrate the meaning of this relationship is with an example.A homemaker has purchases a washing machine from the regional electricity board, as a direct result of which a one year contractual service relationship is established with the board. The nature of the contractual service relationship is one that is defined and enforced by today's legal system. If the washing machine malfunctions in any way the homemaker may invoke the service relationship by engaging in a conversation with the regional electricity board. The purpose of this conversation is to arrange for the malfunctioning washing machine to be fixed. The regional electricity board is bound by the contractual relationship to arrange for the washing machine to be repaired in a timely manner. If the homemaker is dissatisfied with the performance of the regional electricity board in any way then he/she may appeal to the legal system for satisfaction.

Client–Server - In the client–server contractual relationship both parties may engage in the process of service invocation. The client may invoke a service supplied by the server and the server may offer the service to the client without a prior service

invocation from the client. The contractual relationship is used to define not only the nature of the service but also the conditions under which that service may be invoked by the client and offered by the server. An example of such a system is a client server computer system where the client is using a password service that is managed and maintained by the server. Thus the client may request that his copy of the passwords be updated, or the server may force the client to update his/her copy of the passwords.

Consumer–Provider - In the consumer–provider contractual relationship the provider undertakes to provide a service to the consumer, in addition the role of service invoker is something that is performed by the provider. Thus we may say that the consumer–provider contractual relationship is the exact opposite of the consumer–supplier contractual relationship. The consumer–provider contractual relationship is used to define not only the nature of the service but also the conditions under which that service may be invoked by the provider and used by the consumer. A simple example of this relationship that of marriage, as in the marriage contractual relationship the husband undertakes to provide for his wife, and vice versa. The conditions that are placed on this service are that it shall be provided in sickness and in health.

Customer–Supplier - The nature of the customer–supplier contractual relationship is one of either party being able to engage in a conversation with the other party. The purpose of this conversation is to establish another contractual relationship along with the rules governing its existence and meaningfulness. In addition, the nature of the conversation that both parties may engage in is one of negotiation. An example of the customer supplier contractual relationship is one of salesperson–customer. In this relationship the purpose of the conversation is to purchase an item and thereby establish a contractual relationship. This relationship is one that society as a whole defines and consequently the legal system is required to monitor and enforce it. The legal system has over the years established vast volumes of law relating to the governing of this type of relationship. These guidelines govern such things as the permissible and meaningful behaviour of parties that are currently engaged in the relationship. In addition, they also govern the meaningfulness of any contractual relationships that are established as a direct result of this relationship.

5.4.4 The Information Model - The Basic Components

In order to represent the complexity of a social model of information and information processing there is a need for a basic ontology. The purpose of the ontology is to define a language and grammar from which an information model can be constructed. The basic ontology is defined to consist of parties, conversations and information structures.

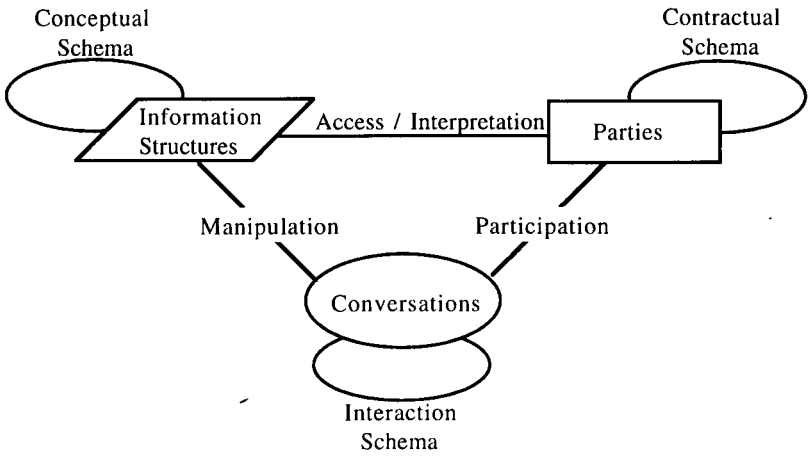


Figure 5.4.2.1 The Information Icon.

This ontology is taken from the conversation level of the information conceptual framework depicted in Figure 3.4.3.1 and is presented in Figure 5.4.2.1. Every concept that was presented in the conceptual framework depicted in Figure 3.4.3.1 is present in the modelling language, and is constructed from the basic ontology presented in this section. For example, a contract is a particular type of relationship that can exist between only two parties.

- **Information Structures**

These are without doubt the easiest and yet the hardest structures to understand. They may be considered to be conventional relational structures on the one hand while at the same time they may also considered to be multi–dimensional structures. For example, we may model information structures in terms of simple relationships (A is_related_to B), or we may model them in terms of relationships between relationships.

- **Conversations**

These are the structures through which by means of their manipulation information structures are created, altered and destroyed. They may be considered to be purely logical structures at the level at which manipulation of the information structures takes place. Each conversation structure is a sequence of manipulations on some information structure that is performed by a party. However the meaning of the conversation is something that requires social concepts in order to interpret correctly. The interpretation of a conversation is something that is derived from the contractual relationships that can exist between various parties. The conversation is by its nature a multi–relational object. For example within an organisation it is meaningless to have a

single party engage in conversation. Conversations are, however, depicted as objects in the information icon, because of the way in which they manipulate the information structure.

- **Parties**

These correspond to the concepts of agents and agency in the enterprise model. For the information model they are considered to be the manipulators and owners of the information. They are the only entities that can perform an interpretation function on the information. They are also the only entities that can participate and engage in conversations. Parties are the entities to which we would prescribe attributes like hopes, fear and intentionality.

- **Conceptual Schema**

The conceptual schema is the set of relationships that can exist between the various information structures.

- **Contractual Schema**

The relationships that can exist between parties are described in terms of contractual relations, e.g. consumer—supplier. The relationships form the social part of the information model in that they give rise and meaning to the set of conversations that two parties may engage in. For example, with a consumer—supplier relationship, one party plays the role of a consumer and the other party plays the role of a supplier. This relationship gives rise to a set of conversations that the two parties may engage in. For example, the consumer party may request a service, and the supplier party is then required to render that service. The set of these relationships that exist between any two parties is called a *contractual schema.*

- **Interaction Schema**

Conversations can refer to each other, and one conversation can be dependant on another. For example, one conversation can be dependant on the successful or unuccessful execution of another conversation. Conversations can also act as a mechanism for structuring interaction. For example a telephone operator may place a caller on hold while the operator engages in another conversation. The set of relations and dependencies between conversations called an *exchange schema.*

- **Access and Interpretation**

Parties can both *access* and *interpret* information. Rules can be expressed over which the parties are allowed to access and interpret information (as opposed to merely accessing it). For example, a secretary who reads a letter on behalf of a principal is certainly to be given access to the information. But an interpretation that the secretary may place upon the information has no validity.

- **Participation**

The participation relationship is used to express the concept of the ways in which a party engages in a conversation. Rules can be expressed to govern the way in which parties may participate in conversations. These rules form part of an organisation's information policy. For example an organisation may have the policy that a computer system can only be accessed between certain hours. Thus a party attempting to access such information may only do so by means of a conversation and such conversations are only valid between certain hours.

- **Manipulation**

The manipulation relationship is the type of relationship that can exist between conversation and information structures in the information icon. Hence we may say that conversations *manipulate* information and information structures. (Manipulation includes the passive case of merely referring to information). This means that information systems also participate in conversations, as when a secretary updates a database.

5.4.5 Modelling Information - An Architectural Approach

In this section the aims and objectives of a particular organisational representation will be presented and discussed along with the organisational representation itself. A representation of an organisation can be constructed from a set of templates where the term template is used to denote an information architecture prefabrication.

The primary purpose of the template, (information architecture prefabrication), as shown in Figure 5.4.5.1 is twofold. The first is to provide a language template from which a set of information diagrams can be constructed, which represent a particular view, perspective, or perception of the system as defined by the information projection. The second is to aid in the construction of an explication of contractual and

conversational relationships by focusing the problem solvers on the conversation objects that exist for a particular party.

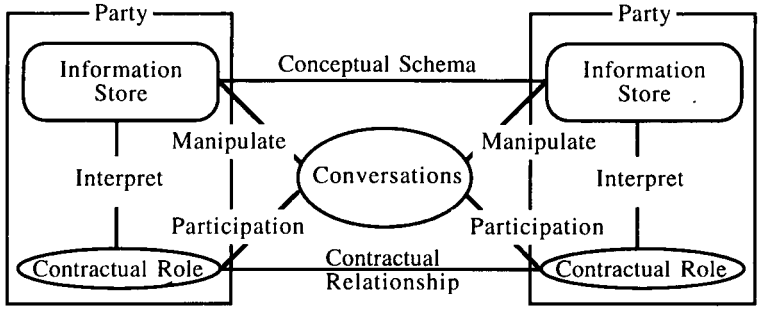


Figure 5.4.5.1 The Information Template

The basic schema that is depicted above can be viewed as consisting of three fundamental building blocks glued together. The first of these blocks is concerned with elucidation of the contractual relationships and roles that are required to exist within the organisation in order for that organisation to function. The second fundamental block is concerned with the way in which people communicate with each other and how information is passed round a social system. The third and final block centres on the information structures and information stores that are required to exist within an organisation in order for the organisation information system to function correctly. Each of these fundamental building blocks will now be examined in more detail.

A contractual relationship links together two contractual roles where each contractual role is held by a party. This basic entity serves to define and give meaning to the expected behaviour that the two parties in the relationship may engage in. From the perspective of the information projection, contracts provide the contextual information that is necessary in order to interpret correctly the semantics of the information model. Consequently the relationship and roles that the two parties may hold act in such a way as to give meaning to the set of conversations that they may engage in. For example, I can engage in a conversation with the bank, the purpose of which is to withdraw some money, by virtue of the fact that I have a particular contractual relationship with the bank.

From the perspective of the information projection, conversations serve as the medium through which an information system becomes a dynamic system. Consequently it is by means of these conversations that the dynamic aspects of an organisational information system can be explored. The types of conversations that can exist within an information system are defined by the types of contractual relationships

that reside within the organisational setting. It is by means of these conversations that the organisational information system is said to function and evolve.

Within the information projection, conversations are used to aid the answering of two types of questions. The first type of question is drawn from the classical world of information modelling, the so-called domain of data engineering. In this domain, the type of question that is addressed is concerned with what information is to be contained within the information system, and how that information is to be manipulated. The second type of question draws its emphasis from the domain of knowledge engineering. This type of question attempts to address some issues concerning how the system, and its environment, will change and evolve over a period of time. In addition, it also attempts to address a class of questions that is much harder to answer concerning the applicability of the information contained within the information system to the organisation's policies and goals.

The third and final fundamental architectural building block of the information modelling template is by far the hardest to conceptualize. This component attempts to address some of the issues concerned with the modelling of the raw organisational information. It achieves this by allowing the concept of information to be examined at two levels. The first level is concerned with the structure, framework and anatomy of an information store, while the second level attempts to address some of the issues concerned with the internal structuring of the information itself.

All three fundamental blocks together form a simple organisational schema from which more complex and detailed organisational information models can be constructed. This simple schema forms the basic architectural unit of the information projection. By drawing on a semi-formalised model of the real world the information architecture can be used to aid in the problem solving process of problems that exist in both the problem and the solution domain.

5.4.6 Representing Organisations - An Information Perspective

A set of information diagrams can be used to model an organisation and thus to express and explore organisational issues and boundaries. A information diagram shows the contractual roles and conversational roles, and the relationships that exist between them. It can therefore be used both as an analytical descriptive diagram, in that it attempts to show the relationships that actually exist within an organisation, and a prescriptive diagram, in that it can be used to represent the required organisational structure.

There are several points that should be noted about Figure 5.4.6.1. Each box, or object, contains a set of conversational and contractual roles linked together by a set of conversational and contractual relationships. The conversational roles and relationships are a shorthand for the types of linguistical behaviours that parties may engage in with one another.

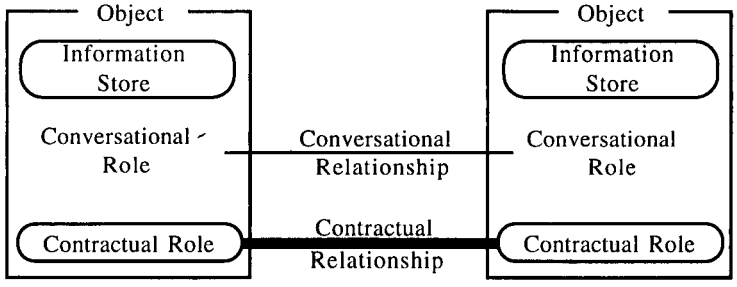


Figure 5.4.6.1 The Basic Model

In addition the contractual roles and relationships are a shorthand for the social contractual framework that permits and gives meaning to these behaviours. Conversational roles are linked by lines of a single thickness, while contractual roles are linked by lines of a double thickness. Within each object is an information store and contained within each information store is a set of information structures. These structures and the information store can only be manipulated by the functional roles and relationships that reside within the object. Together the conversational and contractual roles and relationships and the information stores form an organisational schema defining all aspects of the organisational information system from the perspective of the information projection.

The contractual roles and relationships are used to define and give meaning to the way in which objects within an organisational information system can interact and exchange information. For example, a contractual relationship may define the way in which you can withdraw money from your bank account. The conversational roles and relationships are used to give meaning to and define the interactions that take place between objects in the organisational information system.

The information stores act as the repositories for the information structures. It is by understanding how the information structures and information stores are manipulated over a period of time that an understanding of the organisational information system can be developed and requirements elicited.

Chapter 6 Modelling Organisational Dynamics

6.1 Introduction

"Two things have influenced organisations more than anything else; the coffee machine and the computer. The reason for this is that they both mediate social relationships" (Ehn, 1989).

From this statement I assert that if we are going to develop an information technology system for an organisation we need:

- to understand social relationships and their mediation, and
- to make sure that the information technology system reflects our understanding of social relationships and their mediation.

The first point is addressed within the enterprise and information projections by: a) embedding axiological concepts within the modelling language, and b) defining a modelling language within which we can examine how social relationships are mediated. The second point is addressed by mapping the modelling language used to capture and represent the mediation process down into an executable form.

In the enterprise projection the axiological concept of responsibility is used to define a set of social relationships termed structural relationships. The structural relationships define what resources, and access rights over the resources, are needed in order for a responsibility to be fulfilled. These structural relationships are mediated by means of functional relationships. Functional relationships are a shorthand for the kind of interactions that agent entities may engage in in order to fulfill a responsibility. Structural relationships are a shorthand for the kind of responsibilities

that permit and give meaning to the interactions. It is through these interactions within the enterprise projection that the social relationships are mediated.

In the information projection the axiological concept of contracts is used to define a set of social relationships termed contractual relationships. The structural relationships define what information structures, and access rights over the information structures, are needed in order for a contract to be fulfilled. These contractual relationships are mediated by means of conversational relationships. Conversational relationships are a shorthand for the kind of interactions that parties may engage in in order to fulfill a contract. Contractual relationships are a shorthand for the kind of contracts that permit and give meaning to the interactions. It is through these interactions within the information projection that social relationships are mediated.

Mediation processes are by their very nature dynamic processes. Consequently any modelling language attempting to model the mediation process within an organisation must in fact model the dynamic behaviour of that organisation. In the following sections I will develop a single notation to model the organisational dynamics for both the enterprise and the information projection. I will then show how it is possible to map this notation down into the executable form of coloured Petri nets.

6.2 Modelling Organisational Dynamics

Various notations based on conversation (Auramaki,Lehtinen, & Lyytinen, 1988; Katz & Keshi, 1991), interaction(Auramaki et al., 1988; Finkelstein & Fuks, 1989; Katz & Keshi, 1991), role (Rein & Singh, 1992; Singh, 1992), discourse analysis (Lyytinen,Auramaki, & Hirschheim, 1991) and Petri nets (Beslmuller, 1988; Lee & Sluizer, 1991) have been developed to model organisational information flow and derive organisation and information requirements. These notations have their origins in linguistics (Wunderlich, 1979), philosophy of language (Austin, 1962; Searle, 1979; Searle & Vanderveken, 1985; Wittgenstein, 1958), social theory (Habermas, 1979), role theory (Thomas & Biddle, 1979) and Petri nets (Petri, 1966). They focus on the relationship between the user and organisational environments within which interactions are observed. These interactions are observed as forming a linguistic process the nature of which is the exchange of linguistic utterances between organisational entities. A distinguishing feature of such notations is that they view the processing performed by the Information System as a process of communicative action. Communicative action is carried out through communicative acts, which are the minimal units of human communication.

The problem with notations such as (Davis, 1983; Lyytinen et al., 1991; Singh, 1992) which draw upon linguistic theory to model organisational information systems and problem solving is that while they specify the behaviour of the system, they fail to specify explicitly the organisational structure. The reason for this is that within linguistic theory the social structuring is implicit in the speech pattern and the utterances. Without this organisational structure to compare and contrast the proposed behaviour of the system with, the behavioural models do little more than allow a partial analysis of the organisational information system's behaviour. They do not allow a problem solver to elucidate the functioning of the organisation around the information system.

The notations presented in the following sections will attempt to address some of the deficiencies found in current notations. This will be achieved by drawing upon the social and organisational modelling constructs already presented in this and other chapters. These constructs will be used in such a way as to provide the organisational and semantic means for the interaction to be modelled.

6.3 A Speech Act Model

6.3.1 Introduction

In (Austin, 1962) a simple linguistic model of conversations is presented; this model and related theory became known as speech act theory. Later in (Searle, 1979) more complex and structured objects were added to the model and a logical axiomatic framework was introduced. From within this logical framework the construction, semantics and implications of linguistic interaction can be formally analysed and reasoned about via a model of communication.

The types of questions that the model of communication will attempt to address can be classed into two broad categories. The first category is concerned with the creation and manipulation of responsibilities and obligations in the enterprise projection and the creation and manipulation proportions in the information projection. Through the analysis of such questions we may begin to address such questions as *when does X become responsible for Y.* The second category is concerned with the creation and manipulation of objects such as resources in the enterprise projection and information stores in the information projection.

Let me now illustrate the importance of these two classes of questions with an example. A company cheque acts as a token of both a responsibility and an obligation. The obligation is for the company to pay the amount specified on the

cheque to the person specified on the cheque. The responsibility is for the management of the company to see that the cheque is paid. When an organisation drafts a cheque the interaction notation can be used to aid the elicitation of answers to two distinct questions. The first question is, under what circumstances is the cheque drafted and are all the role holders that either have a responsibility or an obligation placed upon them aware of it. Secondly, is it possible for a role holder not only to be able to write the amount to be paid but also to write to whom the cheque should be paid. Both of these questions are concerned with key organisational policy issues and their answers impact directly on the organisational requirements.

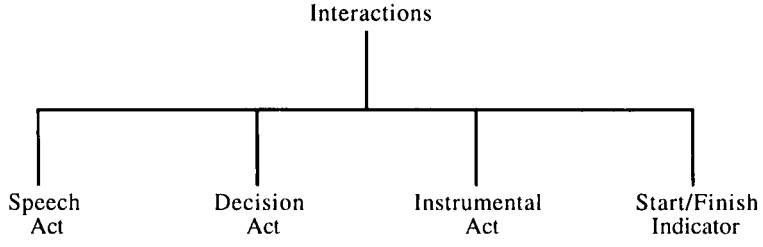


Figure 6.3.1.1 Building Blocks of a conversation.

Figure 6.3.1.1 depicts the basic building blocks from which an interaction diagram can be constructed. Figure 6.3.1.1 shows the basic building blocks of an interaction which are called, from left to right, a speech act, a decision act, an instrumental act and a start/finish indicator. Each of these basic building blocks is now explained in more detail.

6.3.2 Speech Act

6.3.2.1 The Concept

A speech act is a basic unit of communication being a spoken or written utterance that results in meaning being assigned to a linguistic expression (Austin, 1962; Searle, 1969). Speech acts always involve at least two agent/parties roles: speaker and hearer (though these can on occasion be the same individual). Speech acts form larger wholes called *conversations*, which exhibit systematic regularities that can be studied and analysed. An example of a conversation is to authenticate someone; it consists of a number of speech acts such as requesting a token of identification, confirming its authenticity with an authority, and finally accepting the individual as genuine (or not). Speech acts are modelled as taking place over the obligation interface link between structural roles. There are several types of speech acts, of which the three main categories are propositional acts, illocutionary acts, and perlocutionary acts.

A propositional act is a statement which can be evaluated to be true or false, such as "It is Christmas Day today" or "There'll always be an England". Propositional acts can be evaluated to be true or false (though it may not be easy to determine which, as in the second of our examples). We do not say anything more about how the evaluation is performed, nor about the theory of truth we are assuming. Propositional acts can meaningfully be uttered by people or machines.

An illocutionary act, or illocution, is always performed when a person utters certain expressions with an *intention* – for example, "I promise to write the letter" or "I refuse to pay the bill". When the intention has been recognised by the hearer(s), the illocutionary act has been successful, and we say its meaning has been understood. Questions of the truth of an illocution do not arise; rather, the act creates a *commitment* that in a moral sense binds the future behaviour of the parties and pledges them to certain activities or expectations. Illocutionary acts can be expressed through mechanical means as well as vocal means. They are, however, always an expression of human concern or intention.

A perlocutionary act, or perlocution, is an act that produces effects on the feelings, attitudes or behaviour of the hearer(s), for example to get someone to write a letter on request. Again, truth of a perlocutionary act is not an issue; success is, and occurs when the perlocutionary act has its desired effect.

6.3.2.2 The Model

A speech act object aids in the elicitation, comprehension and formalisation of how, when and where organisational attributes such as responsibilities and obligations are either brought into being or fulfilled. A speech act object has four distinct attributes. The first expresses the idea of an agent owning a speech act object, this being the agent that makes the utterance. The second attribute is the type of utterance being made in the object, this in classical speech act theory (Austin, 1962; Searle, 1969) is called the *speech act.* The third attribute is the enterprise, or information, attribute that the utterance manipulates in some way and this attribute is optional. Finally the fourth attribute is the enterprise, or information, object that is accessed in some way as a result of the utterance and this attribute is also optional.

Let me now illustrate how all the above work together to form a speech act object. When a manager gives a command to his subordinate he is making a *directive* type of utterance to the subordinate and thus creating an obligation for the subordinate to execute the command given to him. The obligation is created with reference to the responsibilities that are placed upon the subordinate by virtue of the structural

relationship that exists between the two agents. The utterance acts in such a way as to define a task that the subordinate is required to perform. The supervisor may also create an enterprise object called a specification to aid the subordinate in the execution of his assigned task.

6.3.3 Decision Act

A decision act is used to identify when a role holder has to make a decision of some kind. It is used to allow the problem solvers and problem owners to examine when, where and under what conditions decisions are made. A decision act permits not only the expression of the decision that is being made but also the role holder that is making the decision. In addition conditions that are necessary for the role holder to make the decision can be expressed and explored. An important point to note about the decision act is that the conditions part of the act is optional. The conditions section of the act allows for the representation and examination of the state of the environment that is necessary for the role holder to make a decision. The terms state of the environment is used to encapsulate several concepts. The encapsulation includes concepts such as what enterprise objects and enterprise attributes are required to be present in the system in order for the role holder to make the decision. In addition decision acts aid in the classification and delineation of acts that are affected by, and affect, organisational policy.

The primary purpose of a decision act is to aid the comprehension of what a decision means and what its implications are, i.e. what decisions are critical or not for the fulfillment of the organisation's responsibilities. In addition, decision acts facilitate in the elicitation of what resources are not only necessary, but also sufficient for the role holder to make the decision. The decision act is the only act that combines issues of data flow with control flow and for this reason this act should have special attention paid to it.

6.3.4 Instrumental Act

Instrumental acts not only facilitate the examination and formalisation of when, where and by whom an instrumental act (task) is performed, they also aid the elucidation and identification of what enterprise objects and attributes are required and manipulated by the agent entity performing the act. The expression of the objects and attributes that are required in the performance of an instrumental act is optional and is denoted by a shaded area in the diagram. An instrumental act can either be decomposed down into small instrumental acts or sets of interactions. An example of an instrumental act, is when as will be illustrated in the case study presented in

section 7.2.5, the electrician agent fixes an appliance for the house person agent. In performing this act the agent is not only discharging his obligations but also consuming a resource.

6.3.5 Start/Finish Indicators

The start/finish indicators are used to denote when an interaction begins and when an interaction terminates. they are used purely as control points to denote the entry and exit points of the interaction.

6.4 A Graph Based Model

In Section 6.3.1 a model of interaction is presented which draws upon speech act theory as its underlying formalism. Figure 6.3.1.1 presents the basic constructs from which behavioural and interactional models of the system can be constructed. Each leaf cell on the diagram represents a node in a bidirectional graph that forms the behavioural model. The purpose of the model from the perspective of the enterprise projection is to aid in the elicitation, representation, formalisation and validation of the manipulations and their context that may be performed on enterprise objects and attributes. Consequently, the model would attempt to address such questions as "when, where and under what conditions are these types of enterprise attributes created, modified and destroyed?"

From the perspective of the information projection a different set of projection attributes and objects may be defined. Nevertheless the nature of the questions remains the same. However, the nature of such manipulations on the attributes and objects in the information projection is much more computational and process oriented than the enterprise projection manipulations. For this reason, while the basic diagrammatical notation will remain the same, a more formalistic approach and notation is required. It is for this reason that the same set of nodes that is used in the directed interactional enterprise graph to model and analyse behaviour will also be used in the information projection.

Figure 6.4.1 depicts the more formalistic notation that will be adopted for the information projection. This notation cannot be applied to the enterprise notation as, when constructing such models, the model builders must always keep in the forefront of their minds the questions that such models attempt to address. Within the information projection it is important to note that both data and information engineering is being performed. The enterprise projection serves to aid in the

information engineering part of the information projection by providing a contextual setting within which various informational aspects of the system can be elucidated.

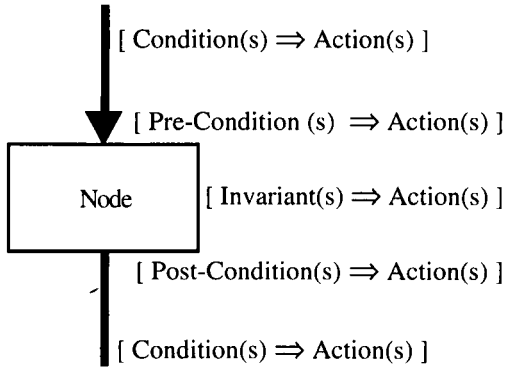


Figure 6.4.1 A Basic Node

While the speech acts, instrumental acts, decision acts, start and stop indicators, and control flow constructs remain the same, a component to express and reason about conditionals and their implications will be introduced into the model. The conditional part of the model is used to express the state of the system and its environment that is required to hold in order for the arcs and nodes to exist, and for the flow of control to pass from one node in the graph to the next. Should no conditional be specified either by an object or by an arc then the default of *true* is assumed. The actions are used to model the way in which the flow of control passing along an arc changes the state of the system. Again as with the conditional if no action is specified then it is assumed that the no state change is visible and consequently the default action is *no_operation*. For example, in a phone conversation by satellite it may take 2 seconds for the utterance made by the speaker to travel to the recipient. Consequently the action on the arc is to increment the time of the system by two seconds.

The pre, post and invariant conditions that are associated with a node on the behavioural graph of a projection serve to define the dependencies that are required to exist within the system in order for a node in the graph to exist and be meaningful, consequential and expressive. The actions that are associated with the pre, post and invariant conditions act to define the way in which the state of the system and its environment change as a result of the execution of the node. For example, a conditional on a node could represent the fact that projection objects of a certain type could only be created after a certain point in time. For example, an organisation may have a policy that all system administration for a computer system is done between the hours of 1am and 3am. Any new users on the system must be generated between those hours. A speech act would be used to represent that fact that a new identifier had been created, while the conditional would be used to depict the temporal and

other dependencies and the action would be used to define the way in which the state of the system changes. For example, it takes 5 minutes to generate a new user on the machine.

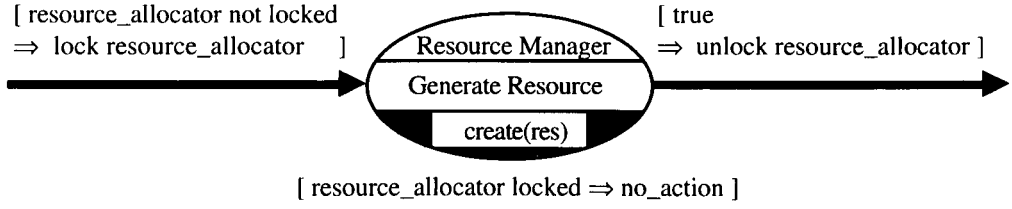


Figure 6.4.2 The Resource Manager Example

Figure 6.4.2 depicts the instrumental act denoting the generation of a new resource. In this example the resource manager is creating and placing a new resource within the system for use. The instrumental act is used to depict and elucidate the linguistic and projectional attributes and objects that are used by the act. The pre, post and invariant condition action statements are used to express the environmental conditions and actions that are required to support and maintain a stable system state. The statements also function so as to identify system objects and the various ways in which the system objects can be manipulated. With use of the condition action statements and the instrumental act diagram, policy statements may be elucidated, modelled and analysed. In addition, this framework provides a means to their implications being explored and analysed.

In the resource manager example, the instrumental act denotes the creation of a new resource. It clearly identifies the nature of the act of creation along with the entity that performs the act. Consequently questions such as "may this entity perform this action?" and, "is this the only entity which may perform this action?" may be expressed and their answers examined. The condition action statements portray the support actions and conditions that are required to maintain the stability of the instrumental act. As a result they define the requirement that in order for the generation act to be performed the resource allocator must be locked and remain locked while the act is being performed. At the end of the act to generate a new resource the resource allocator should be unlocked and thus freed to process requests again. The locking and unlocking of the resource allocator signifies that requests for resources will not, and should not, be dealt with while this action is being performed. The condition action statements also serve to ask to questions of "who, under what conditions and for what purposes can the resource allocator be locked and unlocked?"

6.5 Simple and Coloured Petri Nets

6.5.1 Introduction

Coordination is an elusive concept; if one hundred people were to be asked to define it, then everyone would give a slightly different answer. Even limiting the domain of the problem to organisations the concept still remains an elusive one. To solve this problem, several people have put forward modelling techniques based on Petri nets. In (Beslmuller,- 1988) a model is presented, based on low level place/transition Petri nets, which utilizes functional domain analysis techniques and the OSSAD(Conrath,DeAntonellis, & Simone, 1988) methodology. In this technique Petri net models are created and analysed using standard net analysis methods. Other techniques like (Holt, 1988) attempt to merge some linguistical concepts with the Petri net formalism in order to enlarge the expressive powers of the model. With such a model it is believed that it is possible to capture, express and reason about not only the interaction and cooperation but also the nature and semantics of the interaction and cooperation.

All the current modelling techniques that attempt to analyse organisational interaction and coordination use low level nets as their basic formalism. High level Petri nets such as predicate/transition nets (Genrich, 1986), and coloured nets (Jensen, 1986; Jensen, 1992), have recently been developed and proposed as a superior formalism with which to model, and analyse, organisational interaction and coordination.

The model that will be presented in the following sections will draw upon the coloured Petri net formalism. In using such high level nets to describe organisational interaction and coordination, along with organisational and informational attribute and object life cycles, a higher level understanding of organisational systems issues can be developed and elucidated.

6.5.2 Simple Petri Nets

The simplest Petri net is called a Place Transition (P/T) net and is composed of five parts (P, T, F, W, M_0). The P denotes a set of places, the T denotes a set of transitions and both the union and intersection of P and T is the empty set. The F denotes the flow relation function — this function is the union of two cartesian products. The first cartesian product expresses the flow from places to transitions, while the second expresses the flow from transitions to places. The W denotes the weight function, this function for a given arc tells how many tokens may flow down

that arc at once. Finally M_0 expresses the initial marking of the net — this tells us what tokens are in which places to begin with. A transition without an input place is called a source transition and one without any output transition is called a sink transition.

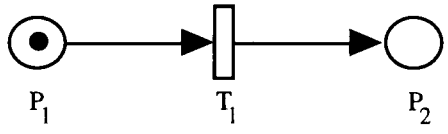


Figure 6.5.2.1 A Simple P/T Net

Figure 6.5.2.1 is an example of a very simple place transition net. The circles are used to denote places and the rectangle is used to denote a transition. Figure 6.5.2.1 consists of two places called P_1 and P_2 and a transition called T_1. The names of the places and transitions are meaningless to the formalism and are purely used to aid in the comprehension of the net in solving a particular problem or modelling a particular system. The arcs from places to transitions and transitions to places are used to express the flow of tokens from one place to another via a transition. Transitions within Petri nets are viewed as system state changing operators. Consequently there is a simple syntax for these nets: that places are connected together with arcs that go via transitions and conversely transitions are connected together with arcs that go via places. The black dot shown in place P_1 in Figure 6.5.2.1 is a token and a net is executed via the movement of tokens from place to places via transitions.

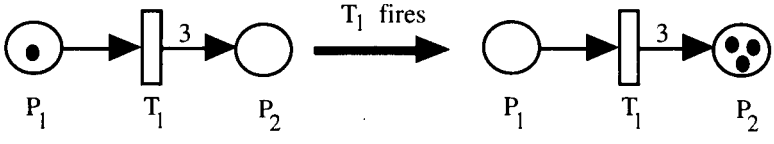


Figure 6.5.2.2 The Execution of a Net

Figure 6.5.2.2 expresses the execution of a net via the firing of transition T_1, thus the state of the net changes from the net depicted on the left to the net on the right. The diagram also illustrates the weight of an arc, this is shown as the number 3 above the arc from transition T_1 to place P_2. The meaning of this is that when transition T_1 fires, one token is taken from P_1 and two tokens are placed in P_2. A transition can only fire when all of its input arcs are said to be activated. This is when there are enough tokens in all of the input places to satisfy the weighting of the arcs. If no weight is depicted above an arc then the default of one is used.

Figure 6.5.2.3 shows the two possible firings that can occur for the net depicted in the top left hand corner of the diagram. The possibility of two firings

arises as the place P_1 has two arcs originating from it and going to the transitions T1 and T2. As a token can only traverse down one arc at once, a choice has to be made as to which arc the token will travel down. For P/T nets this choice is non–deterministic in nature and from the perspective of the observer is random. For other high level nets such as coloured nets which will be discussed later, conditions can be placed on arcs to overcome this problem, thus making the choice deterministic.

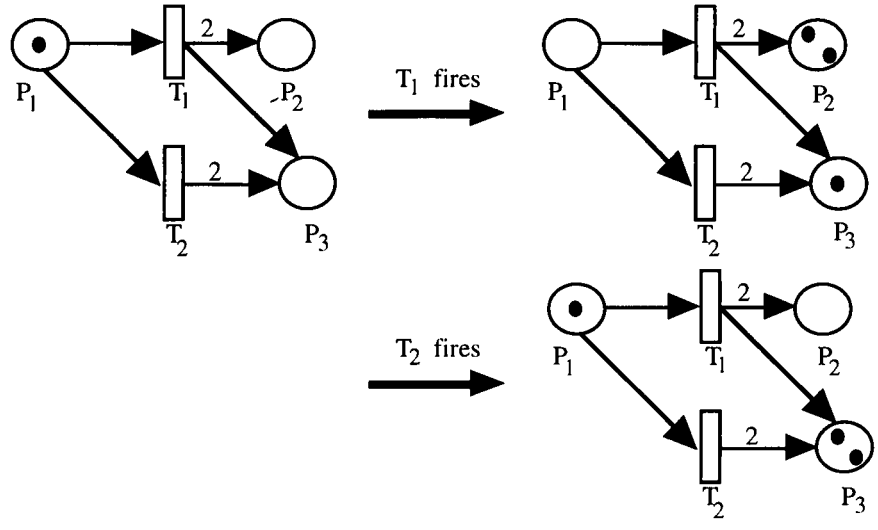


Figure 6.5.2.3 Executing a Complex Net.

6.5.3 Coloured Petri Nets

The hierarchical coloured Petri nets (CP Nets) presented in this section and used within this thesis are taken from (Jensen, 1992). However it should be noted that the work presented in (Jensen, 1992) builds upon the non-hierarchical coloured Petri nets that were first presented in (Jensen, 1986). A predecessor and competitor to the CP nets notation is that of predicate transition nets (P/T Nets) (Genrich, 1986). This notation is based upon the concept of relations and relationships as opposed to the functional notation from which CP nets draw their semantics. From a purely descriptive level both nets can be said to have the same descriptive powers. However the the CP nets have a wider selection of tools to aid in their construction and analysis. Coloured nets have for this reason been more generally adopted by industry and it is for this reason that they will be adopted and used in this thesis. One of the most important properties of CP nets is that, in contrast to many other graphical description languages, they have well defined semantics which in an unambiguous way define the behaviour of the system.

Figure 6.5.3.1 is a example of a simple non–hierarchical coloured Petri net and there are several points that should be noted about the notation and diagram. This diagram is a simple representation of a small resource consuming system. As with the

simple Petri nets presented in section 6.5.3.2 of this chapter the circles represent places, the rectangles represent transitions and the arcs represent the mapping of one to the other. Unlike the simple Petri nets illustrated earlier the CP tokens are complex hierarchical data structures where the data structures associated with a token are referred to as the *token's colour.*

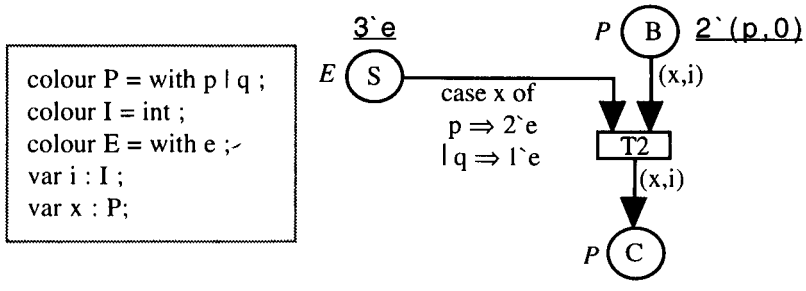


Figure 6.5.3.1 A Simple Coloured Net

A coloured net is said to consist of three distinct parts: the *net structure*, the *declarations* and the *net inscriptions* . The net structure is a direct graph consisting of two types of nodes, places and transitions, interconnected by arcs. The directed graph is connected together in such a way that each arc connects exactly two different types of nodes, i.e. a place and a transition, and such graphs are called bipartite directed graphs. The declarations in the left hand side of Figure 6.5.3.1 tell us that in this simple example there are three colour sets (P,I and E) and two variables (i and x). In this example the set P contains two elements p and q and the set I is the set of all integers, whilst the set E consists of only a single element e. A net inscription can be attached to either a place, a transition or an arc. In Figure 6.5.3.1 the places are said to have three different types of inscription: *names*, *colours sets* and *initialization expressions.*

The transitions are said to have two different types of inscriptions: *names* and *guards*, and arcs are said to possess only a single kind of inscription: *arc expressions.* All net inscriptions are positioned next to the corresponding net element. In addition, to make it easy to distinguish between them, names are written in plain text while the colour sets are denoted by italics. Initialization expressions are underlined and guards are contained in square brackets.

Names have no formal meaning and they purely serve as a means of identification and delineation. The initialization expression of a place must evaluate to a multi-set over the corresponding colour set. Multi-sets are analogous to sets except that they may contain multiple appearances of the same element. In the case of CP nets, this implies that two tokens on the same place may have identical colours.

The convention is that we omit initialization expressions which evaluate to the empty set.

The guard of a transition is an expression which evaluates to true or false and must be fulfilled before the transition can occur. The convention is that guards which under all conditions evaluate to true are omitted. The arc expression of an arc is an expression, and it may contain variables, constants, functions and operators that are defined in the declarations (implicitly or explicitly). When the variables of an arc expression are bound then the arc expression must evaluate to a colour that belongs to the colour set attached to the place of the arc.

A distribution of tokens is called a *marking* and is denoted by the letter *M*. The *initial marking* is the marking determined by evaluating the initialization expressions and is denoted by the term M_0. The *occurrence element* denoted by the term O_n. is a pair, where the first element is a transition and the second element is the binding of that transition. We may now begin to analyse the net by asking whether an occurrence element O_1 is enabled in a given marking M_1 and when this is the case then we may say that the marking M_2 is *reached* by the occurrence of O_1 in M_1.

It is important to note that several occurrence elements may be enabled in the same marking. If that is the case then we may say that either there are enough tokens so that all the occurrence elements can be activated, or there are not enough tokens for all the occurrence elements to be activated. If all the occurrence elements can be activated then we may say that the occurrence elements are *concurrently enabled.* However if all the occurrence elements cannot be activated then we say that they are in *conflict* with each other. The definitions given above are an example of one possible way in which a net may be analysed, for a complete set of formal definitions and analysis techniques (Jensen, 1992).

The hierarchical CP nets presented in (Jensen, 1986) comprised the first successful attempt to create a set of hierarchy concepts for a class of high level Petri nets. Individual CP nets with the hierarchical coloured Petri nets framework are known as pages. Within the hierarchical coloured Petri nets framework there are five possible ways of constructing a hierarchical CP net each of which is called a *hierarchy construct.*

These are substitution of transitions, substitution of places, invocation of transitions, fusion of places and fusion of transitions. The idea behind substitution of places and transitions is to replace a place or transition with a more complex CP net.

This idea is analogous to the hierarchy constructs found in many graphical description languages e.g. SADT (Ross & Schoman, 1977).

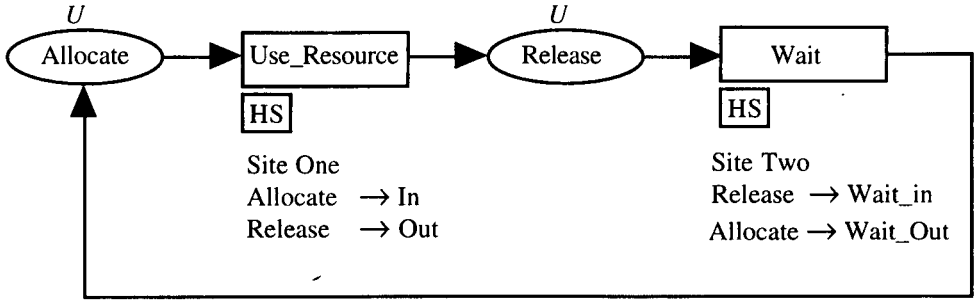


Figure 6.5.3.2 A Hierarchical Net.

Figure 6.5.3.2 is a simple example of a hierarchical net that describes a resource allocation system. In this example resources are requested and allocated - these resources are then used in some manner and finally released back to the resource management system. In Figure 6.5.3.2 the transition *use_resource* has a HS tag (HS = Hierarchy + Substitution). The inscription next to the HS tag is known as the *hierarchy inscription*, and it defines the details of the actual substitution.

The first line of each hierarchy inscription identifies the *subpage*, i.e. the page that is going to replace the substitution transition. The other lines of the hierarchy inscription contain port assignments, which tell us how the *subpage* is going to be inserted into the *superpage*. Each line of a port assignments defines how a *socket node* on the superpage relates to a *port number* on the subpage and how the substitution of transitions will occur.

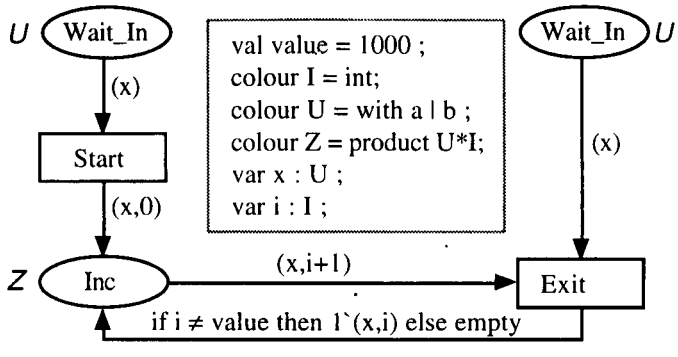


Figure 6.5.3.3 Site Two

Figure 6.5.3.3 is a simple delay loop the symbolism being that of waiting for a specified amount of time. One of the problems of coloured nets is that they have no concept of time other than that which the user builds into the model. Figure 6.5.3.4 has two transitions symbolising the beginning and terminating of the delay loop and three places symbolising the entering into, the execution of and the exiting from the

delay loop. When Figures 6.5.3.2, 6.5.3.3 and Figure 6.5.3.4 are merged together the coloured net shown in Figure 6.5.3.5 is produced.

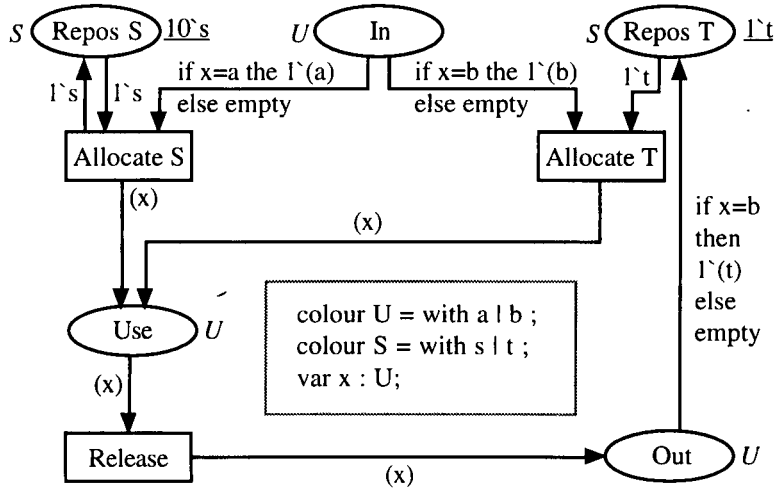


Figure 6.5.3.4 Site One

Figure 6.5.3.4 is a pictorial representation of a simple two type of resource management system. In this diagram there are two types of resources *s* and *t* and two types of requests *a* and *b*. In addition there are three transitions representing the allocation and release of resources and five types of places representing the arrival of a request; the repository for resources of type s; the repository for resources of type t, the use of a resource and finally the message passing out of the resource management system. The colour *U* represents the two types of requests that can arrive for the different resources. The colour S represents the two different types of resources and the variable x represents a message.

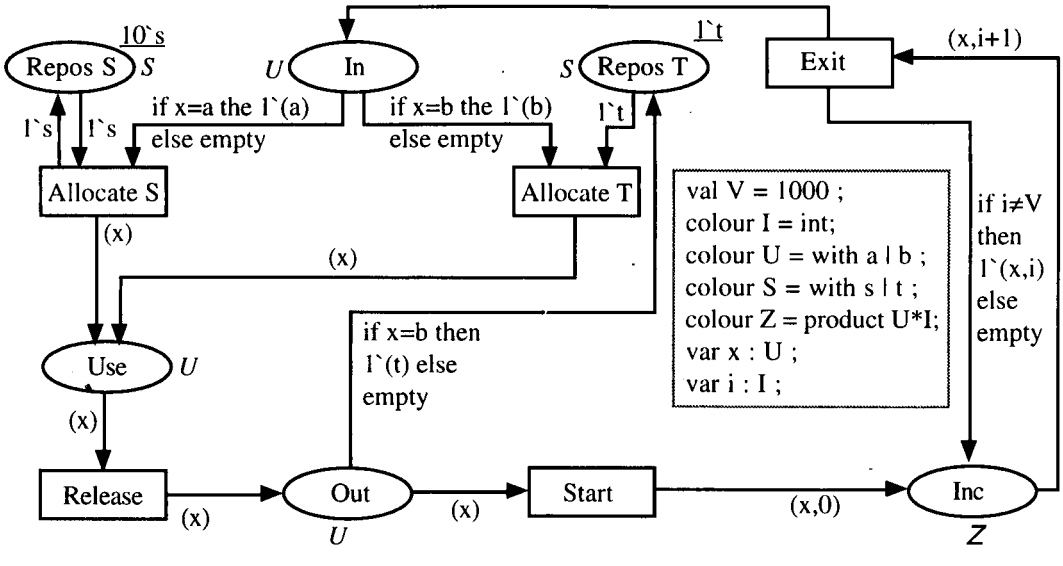


Figure 6.5.3.5 The Complete Picture

In Figure 6.5.3.5 the semantics and behaviour are the same as Figure 6.5.3.2 - in all facets they are the same.

6.6 A Coloured Petri Net Model of Interactions and Conversations

6.6.1 Introduction

In this and the following sections a coloured Petri net model is developed to facilitate the examination, elucidation and analysis of the dynamic aspects of both the enterprise and the information projections. In the enterprise projection the concept of interaction is developed to model and analyse the behavioural aspects of the organisational system. With the information projection the concept of conversations is used to model and analyse the behavioural aspects of the information system. For this thesis a coloured Petri net model will be used as the underpinning representation and analysis formalism for both interactions and conversations. The concept of time will be introduced so that the temporal aspects of the model may be expanded upon, formalised and finally analysed. Consequently both projections may ask and answer question such as *when is this entity created* or *when is this relationship used?*

6.6.2 The Coloured Sets and Some Basic Functions

Each CP net has a set of declarations, which by convention are positioned inside a dashed box. These declarations introduce a number of colour sets, functions, operators, variables and constants each of which may then be used in the guards, arc expressions and initialisation expressions of the coloured net. The standard language that is used to define the functions, operators, variables and constants is called CPN ML. This language is based upon, and derived from, the functional programming language called Standard ML.

In order to define fully the coloured sets that will be required to analyse the Petri model of interactions and conversations a set of functions needs to be defined. The first function that will be defined is called *date*; it takes no arguments and returns the current date in a data type of the form *day:month:year*. The operators that may be applied to this new data type are assignment =, comparison ==, addition + and subtraction -. The second function is called *time* and returns the current time of the system in the form hours:minutes:seconds, where $0 \leq$ hours < 24, $0 \leq$ minutes < 60 and $0 \leq$ seconds < 60 and hours and minutes are integers and seconds are real. Again as with the previous data type the operators that we may wish to apply to the new data

type are assignment =, comparison ==, addition + and subtraction -. For example we ask and answer the question what is the date 20 days and 1 year from now simply by writing *data() + 20:00:01*. In addition we may also ask and answer the question what is the time 23 hours, 14 minutes and 25.65 seconds from now by writing *time() + 23:14:25.65*.

Both the information and the enterprise projections will have their own set of coloured sets – however for the purpose of this thesis, only the coloured sets for the first two projections (the enterprise and information) will be presented. Each of these two projections define their own set of questions along with their own models, modelling languages, objects and attributes that attempt to answer, or elucidate on the nature of, the questions. The function *projection()* is used to determine within which projection the Petri net model is being used. If the model was being used within the enterprise projection, the *projection()* function would return the value enterprise and if the model was being used within the information projection then the function would return the value information.

However before the coloured sets that will be used to analyse the enterprise and information projections are presented some basic types need to be defined. The term *Enterprise_Actions* (EA) will be used to denote the set of allowable manipulations and actions that may be performed on an enterprise object. For example Enterprise_Action = {read, write, create, destroy, consume, etc.}. In addition the term *Enterprise_Objects* (EO) will be used to denote the set of allowable enterprise objects. For example Enterprise_Objects = {specification, nuts_and_bolts, etc.}. The term Enterprise_Manipulations (EM) will be used to denote the set of allowable enterprise attribute manipulations. For example, Enterprise_Manipulations = {create, destroy, amend, etc.}. Finally the term Enterprise_Attribute (ET) will be used to denote the set of allowable and meaningful enterprise attributes.

The term *Information_Action* (IA) will be used to denote the set of allowable manipulations and actions that may be performed on an information object. For example, Information_Action = {read, write, create, destroy, consume, alter, etc.}. In addition, the term *Information_Object* (IO) will be used to denote the set of allowable and meaningful information objects. For example Information_Object = {specification, identifier, password, etc.}. The term Information_Manipulation (IM) will be used to denote the set of allowable information attribute manipulations. For example, Information_Manipulations = {create, delete, amend, alter, etc.}. Finally the term *Information_Attribute* (IT) will be used to denote the set of allowable and meaningful information attributes. The term Agent is used to denote the set of all agents that may exist within the enterprise projection. The term Party is used to

denote the set of all parties that may exist within the information projection. Finally the term Sp_Act is used to denote the set of all speech act utterances that may exist within both projections. The coloured sets that are to be used by the Petri net model to analyse the dynamic part of the enterprise projection are defined as follows :

```
Colour EOC =    Product EA * EO                    ;
Colour EAC =    Product EM * ET                    ;
Colour EP  =    Record
                        p  :  Agent                *
                        s  :  Sp_Act               *
                        e  :  set of EOC           *
                        a  :  set of EAC           *
                        d  :  date                 *
                        t  :  time                 ;
Colour CEP =    set of EP                          ;
Colour IOC =    Product IA * IO                    ;
Colour IAC =    Product IM * IT                    ;
Colour IP  =    Record
                        p  :  Party                *
                        s  :  Sp_Act               *
                        e  :  set of IOC           *
                        a  :  set of IAC           *
                        d  :  date                 *
                        t  :  time                 ;
Colour CIP =    set of IP                          ;
Colour ITT =    Product CEP * CIP                  ;
```

Figure 6.6.2.1 The Colour Sets

Figure 6.6.2.1 represents the set of colours that are required in order to construct and analyse correctly the coloured Petri net model of the enterprise projection. Questions like "when is this enterprise object accessed?", "when is this enterprise attribute constructed?" and "when is this speech act made?" can all be addressed. In addition, Figure 6.6.2.1 also represents the set of colours that are required in order to construct and analyse correctly the coloured Petri net model of the information projection. Questions like "when is this information object accessed?", "when is this information attribute constructed?" and "when is this speech act made?" can all be addressed. At every place and at every transition in the execution of a coloured net the token is updated to say where it is, and what the current state of the environment is. In effect the token has a history, it remembers where it came from and how it got here. Through various executions of the net we can analyse the token and answer the questions defined above.

In the following sections the mappings will be presented from the interaction and conversation models via the coloured hierarchy net to a low level coloured Petri net model. This mapping via a hierarchy net simplifies the mapping process and allows for an intermediate validation step to be introduced so that the problem solvers

and problem owners can be sure that they have captured and understood the nature of the interaction. The mapping process becomes one of transcription and subsequent validation, after which the transition substitution can be performed and analysed before representation and validation is performed back to the problem solvers.

In the mapping from the interaction and conversation notation to the hierarchy net the conditional action statements are simply mapped into comment statements that are associated with the hierarchy inscriptions. When the transition substitutions are performed the conditional action statements are mapped down into the formal net language. It is this mapping that has to be performed in an interactive manner between the problem solvers and the problem owners. Thus when the final net is produced and analysed the problem owners may have confidence that they have captured and modelled the correct set of interactions.

6.6.3 A Coloured Net Model of the Start/Finish Indicator Constructs

The start and finish indicator constructs are used as binding points for the beginning and termination of a set of linguistic interactions. Their primary purpose is to aid in the elucidation of the conditions surrounding the initiation and termination of the interaction. The Petri net model provides a more formal framework within which it is possible to express and reason about such concerns.

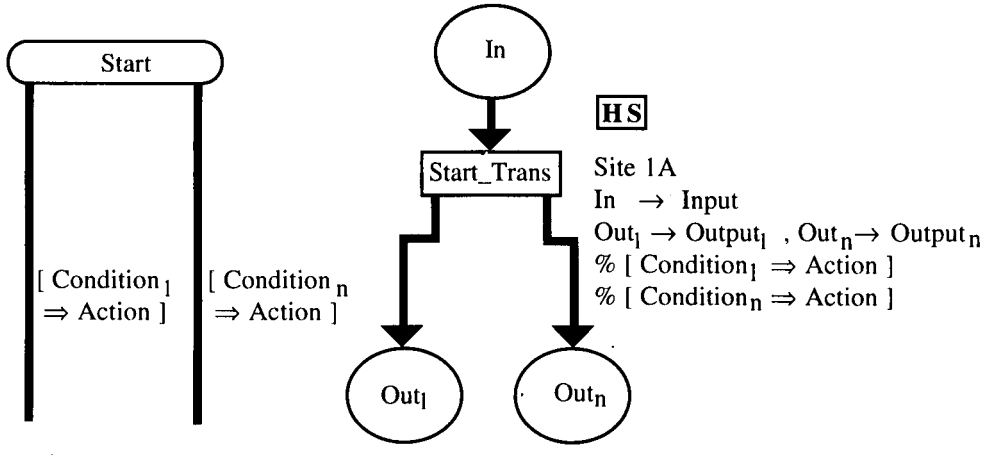


Figure 6.6.3.1 The Start Interaction Petri Net SuperPage Model

Figure 6.6.3.1 represents the mapping from the interaction start indicator to its hierarchy net equivalent. In this diagram the top illustration is the graphical notation denoting the start indicator of the interaction or conversation. The bottom illustration describes the hierarchy net equivalent of the start indicator construct, and such nets are called superpages.

Within the hierarchy net the HS inscription denotes the hierarchy and substitution tag while the text next to this is called hierarchy inscription. Within the hierarchy inscription there is a unique site number which gives us the subpage of the mapping. It is this subpage that is used in the identification and transition substitution of the low level coloured net. The next line gives the port assignment which describes the interface between the subpage and the superpage. The final line in the tag starts with a % and this identifies it as a comment. These comments are used to map down the high level informal condition action statements into the informal condition action statements contained in the low level coloured net model.

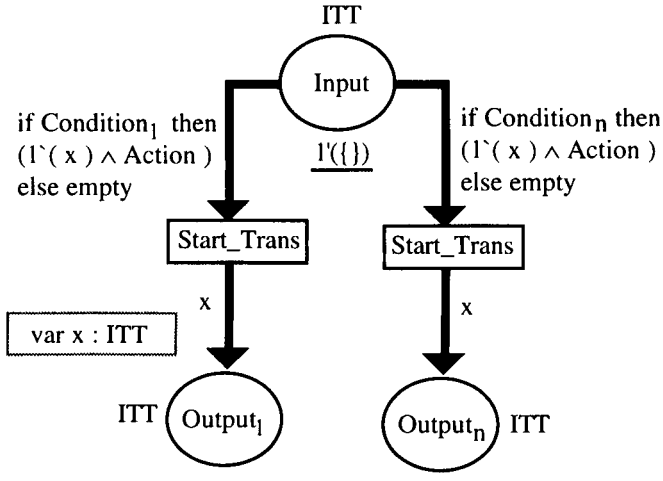


Figure 6.6.3.2 Site 1A The SubPage Model of the Start Indicator

In Figure 6.6.3.2 the transition substitution for the interactions and conversations start indicator is depicted. In this diagram the information that is contained in the hierarchy inscription is mapped down into the subpage (low level) coloured net model. The informal condition action statements that are contained in the interaction and conversation models are now mapped down into real predicate conditions and actions. It is the mapping process of taking the informal predicates and mapping them to formal predicates which needs to be performed by the problem solvers in an interactive and iterative manner.

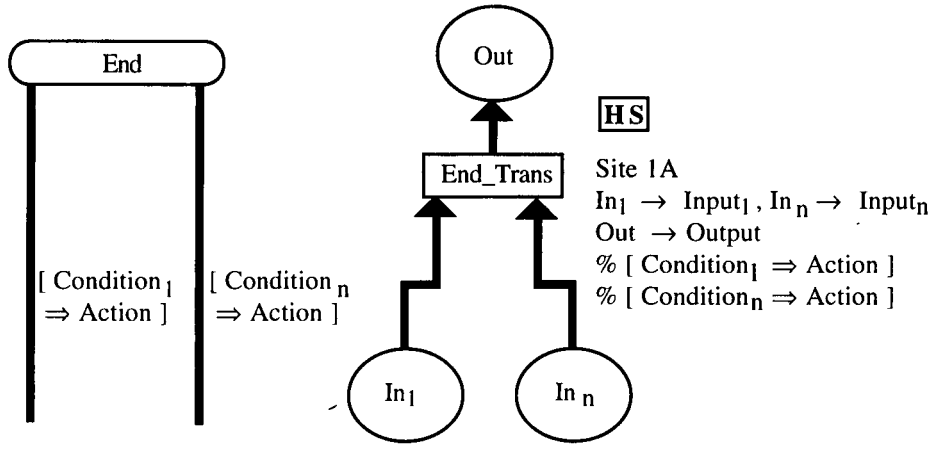


Figure 6.6.3.3 The End Interaction Petri Net SuperPage Model

Figure 6.6.3.3 represents the mapping from the interaction termination indicator for conversations and interactions to its hierarchy net equivalent. The mapping is much the same as that of the start indicator as all the inscriptions have the same meaning. In Figure 6.6.3.4 the transition substitution of the start indicator for interactions and conversations is depicted.

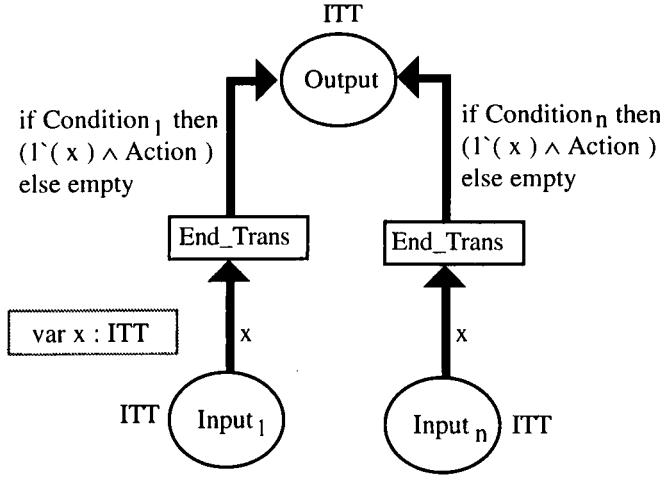


Figure 6.6.3.4 SITE 1B The SubPage Model of the End Indicator.

6.6.4 A Coloured Net Model of the Speech Act Construct

The speech act model construct is used to analyse the interactional component of the information and enterprise behavioural models. A speech act is a basic unit of communication (Austin, 1962; Searle, 1969), this communication can take the form of verbal or nonverbal interaction. We use the model in the information and enterprise projections to construct and analyse behavioural models of the system that capture the type and nature of interactions that can exist between the various agents or parties that exist within the projection models of the organisation. So for example, using speech

acts we can answer questions such as "who makes this speech act ?" and "who creates this responsibility ?" The Petri net speech act model is used to analyse and examine various formal aspects of the information and enterprise behavioural models. For example, using Petri nets we can answer questions such as "does this speech act ever get made ?" and "do we ever get into an infinite loop ?"

The actions derived from the speech act conditionals are used to model the way in which the system's state may change as a result of the component's execution. One of the purposes of the Petri net model is to analyse the more state transitional component of the linguistic behavioural model. Questions like "does this part of the conversation ever get executed?", or policy statements like "all information of this type will have been destroyed by this point in time", can be formally reasoned about and their implications formally and explicitly stated.

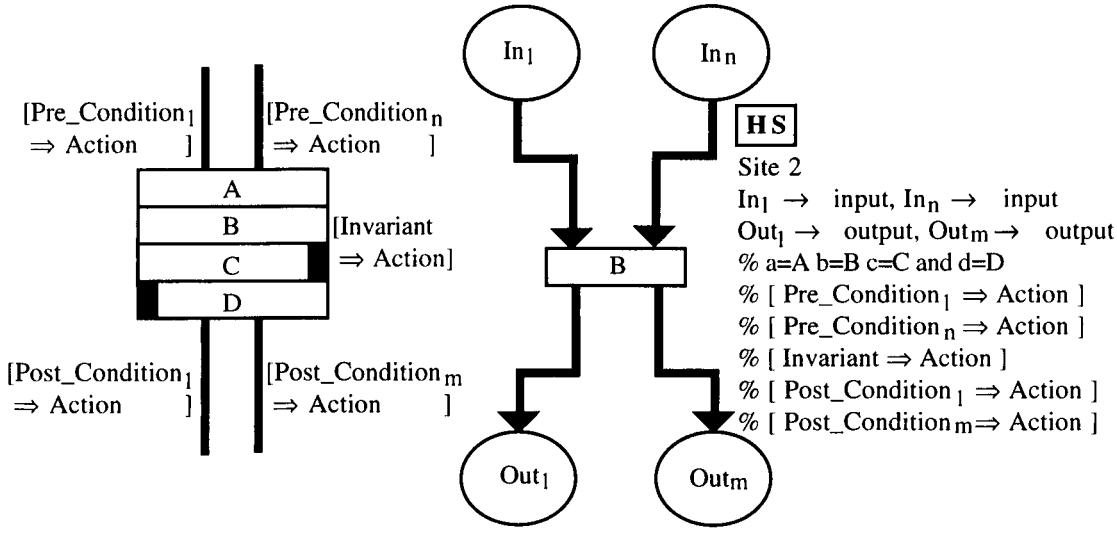


Figure 6.6.4.1 The Speech Act Petri Net SuperPage Model

Figure 6.6.4.1 depicts the mapping from the interactional and conversational model of behaviour to the hierarchy net model of behaviour. The actions that may be inferred from the conditionals attached to various parts of the speech act are now mapped down into the comment part of the hierarchy inscription. The pre-condition, post-condition and invariant are all mapped down and commented into the hierarchy inscription, along with the various components of the speech act. The utterance depicted in the speech act model is used to name the transition of the Petri net model. This has no formal implications as it adds solely to the user's comprehension and understanding of the net. In a colour net, each place and transition is unique in is own right. This uniqueness of these objects stems from their existence and not their names. Names have no formal meaning and they purely serve as a means of identification and delineation.

In Figure 6.6.4.2 the transition substitution of the speech act to its sub page coloured net is depicted. In this diagram all of the formal condition action statements are derived from the information condition action statements contained in the speech act. For the pre and post conditions, the correct execution of the conditional results in the invocation of the action statement. However, the correct execution of the invariant implies not only the execution of the action statement, but also the amendment of a token to indicate that a speech act has been made. This amendment takes the form of appending the set contained within the token with the details of the speech act.

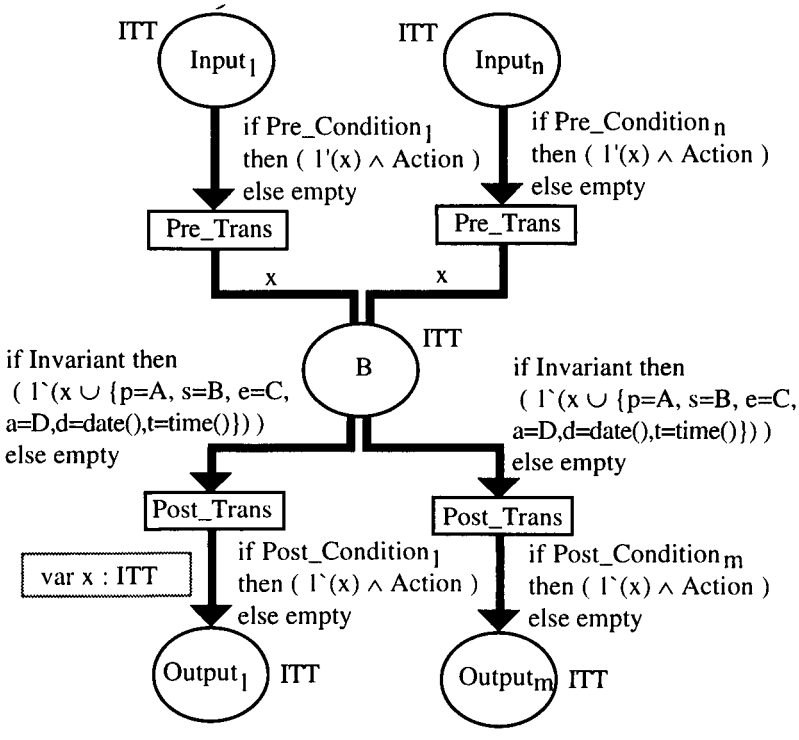


Figure 6.6.4.2 SITE 2 The SubPage Model of the Speech Act.

6.6.5 A Coloured Net Model of the Decision Act Construct

The decision act construct is used to analyse not only the various aspects of how, when and where decisions are made, but also to elucidate the nature of the decision. The Petri net decision act superpage model is used to analyse the various formal aspects of the decision itself. The actions that are derived from the pre and invariant conditions are used to model the way in which the system state changes as a result of the construct's execution. The mapping down of the condition action statements from the high level speech act notation to the low level coloured net notation (via the net hierarchy) provides a framework from within which the nature, and implications, of the decision may be explored and examined. The Petri net framework provides a formal framework from within which questions like, "does that decision ever get made?" and "what projection objects and attributes are necessary

and sufficient for this decision to be made?". The nature of the decision is used to name the place and transitions that denote the decision act in the Petri net model. This has no formal implications, other than that it aids the user of the notation in the comprehension of the model.

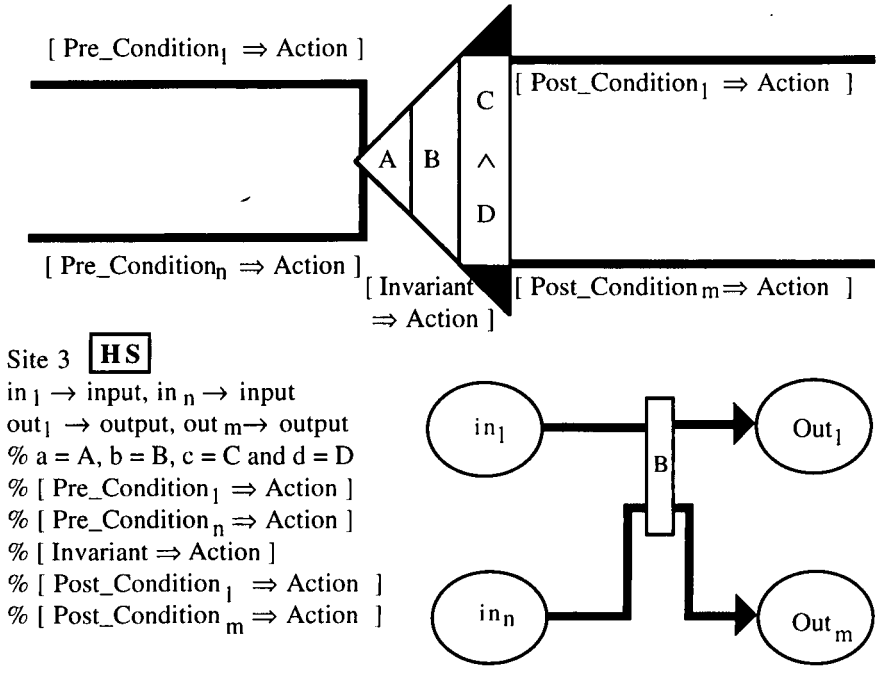


Figure 6.6.5.1 The Decision Act Petri Net SuperPage Model

Figure 6.6.5.1 depicts the mapping from the decision act model of behaviour to the superpage of a Petri net hierarchy model. The speech act is depicted in the top of the illustration, while the superpage is depicted in the bottom of the illustration. The Petri net model is used to analyse the various formal aspects of the behavioural model. The actions that may be inferred from the conditionals placed on arcs in the behavioural projection model are now mapped down into the comment component of the hierarchy inscription. The decision act is used to express in an informal manner that a decision is to be made at this point in the conversation or interaction. Thus the notation aids in the representation and elucidation of the nature and requirements of the decision.

In Figure 6.6.5.2 the sub page low level Petri net model is depicted. In this diagram the informal condition actions of the decision act diagram have been mapped down into formal statements which can be validated against the state of the organisational environment. Again as with the speech act, the pre and post condition action statements are mapped down so that when the conditional executes correctly then the action statement is performed. When the invariant conditional is executed correctly then the token is updated to indicate that the decision has been made. The

post conditions that are associated with the arc emanating from the decision act are not to say what the outcome of the decision should be, but rather to say whether or not that particular outcome may be considered a valid outcome.

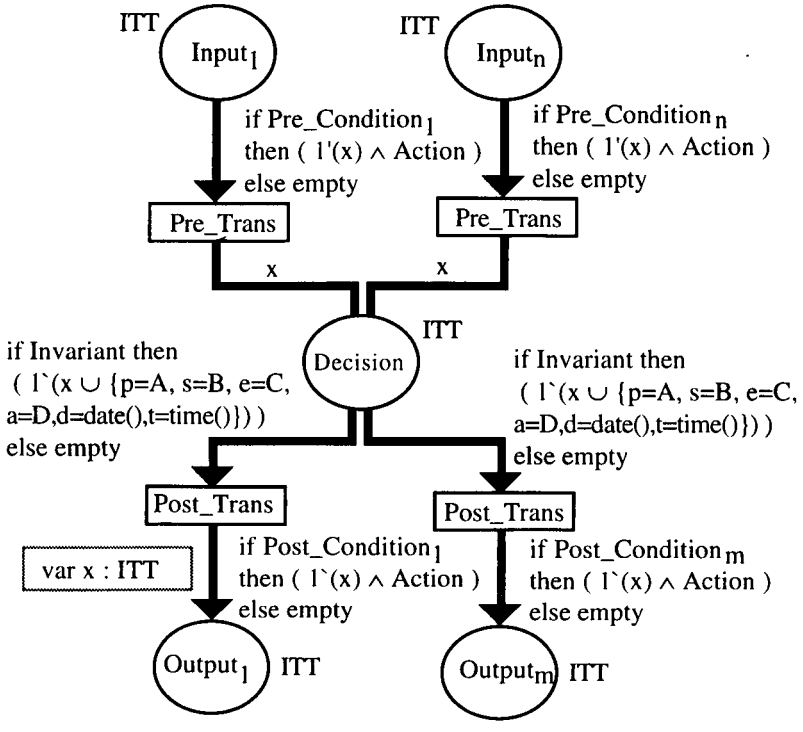


Figure 6.6.5.2 Site 3 The SubPage Model of the Decision Act

For example, when logging into a computer system it may be a requirement that you can only have three attempts to log in before the computer freezes your account. Thus the conditional on the arc would say that the freeze account option was only valid on the third attempt to log in. The decision act does not contain any explicit rules governing the making of the decision. It will merely state what conditions are required in order for a decision to be made. It should be noted that in Figures 6.6.5.2 and 6.6.5.2 the term C is used to denote any actions and the objects upon which the actions are to be performed. In addition, the term D is used to express any manipulations, and the projection attributes upon which the manipulations are to be performed.

6.6.6 A Coloured Net Model of the Instrumental Act Construct

The instrumental act model construct is used to analyse the linguistic component of the information and enterprise behavioural models. The Petri net instrumental act model is used to analyse the more formal aspects of the information behavioural and enterprise behavioural models. The actions that are implied by the conditional associated with the instrumental act are used to model the state changing

behavioural aspects of the Petri net model. From the Petri net model such aspects as temporality and functionality can be analysed within a formal and structured framework. The formal analysis techniques that have been developed to analyse these state transition models can be used to derive requirements and this feed back into the information and behavioural models.

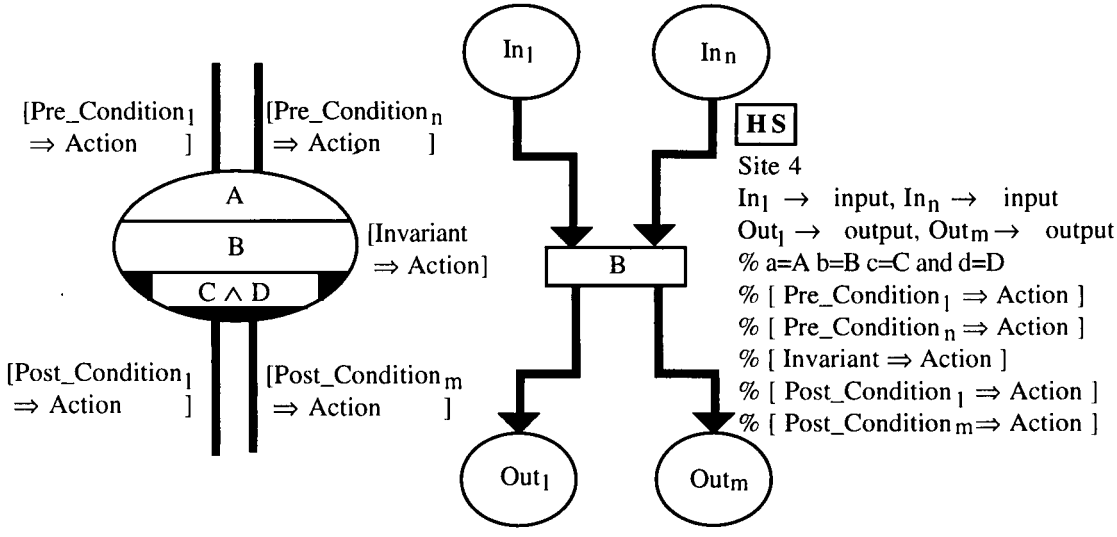


Figure 6.6.6.1 The Instrumental Act Petri Net SuperPage Model

Figure 6.6.6.1 depicts the mapping from an instrumental act behavioural model to a superpage hierarchy Petri net model. The informal condition actions statements that exist on the instrumental act are mapped down into the comment component of the hierarchy inscription of the superpage model. In addition, the various components of the instrumental act are also placed in the comment component of the hierarchy inscription.

In Figure 6.6.6.2 the sub page coloured Petri net model of the instrumental act is depicted. In this diagram the informal condition action statements of the instrumental act have been mapped down into formal condition action statements. If the pre and post condition action statements hold at the time of their execution then the action statements are performed. The holding of the invariant conditional at its time of execution however also implies that the token is updated to indicate that the instrumental act has been executed. This is achieved through the various components of the instrumental act model being mapped down into the token. The term C is used to denote the set of actions that are to be performed on various projection objects along with the projection objects themselves. The term D is used to express the set of manipulations that are to be performed on various projection attributes along with the various projection attributes.

The utterance depicted in the instrumental act model is used to name the place and transition of the Petri net model. This has no formal implications as it adds solely to the user's comprehension and understanding of the net. The invariant conditional is also placed on the transition that denotes the execution of the instrumental act to act as a guard. The guards are boolean expressions that must be fulfilled before the transition can occur.

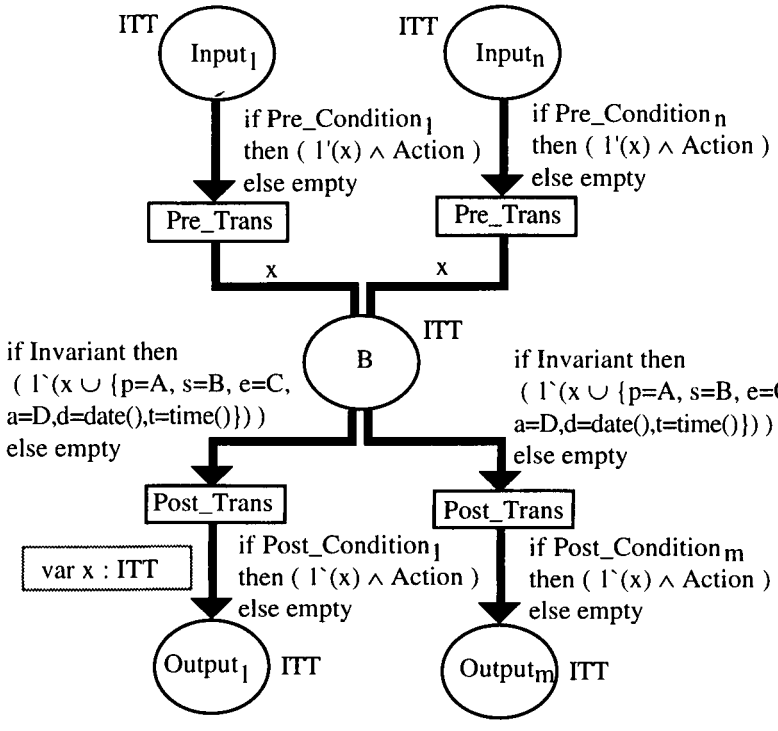


Figure 6.6.6.2 Site 4 The SubPage Model of the Instrumental Act

6.6.7 A Coloured Net Model of the Arc Construct

The arcs are used to represent the flow of control and the exchange of information from one node on a conversation diagram to the next. They are used to express the concepts of interactions travelling, via some medium, from one entity to the next. The conditional action statements are used to express the notion of the flow of communication changing the system state. For example a piece of communication may take 5 seconds to travel from one entity to the next.

Figure 6.6.7.1 depicts the mapping from the conversational flow of a control behavioural model to a coloured Petri net state transition model. The conditional and action implied from that conditional is placed on the arc between the transition and the place in the Petri net model.

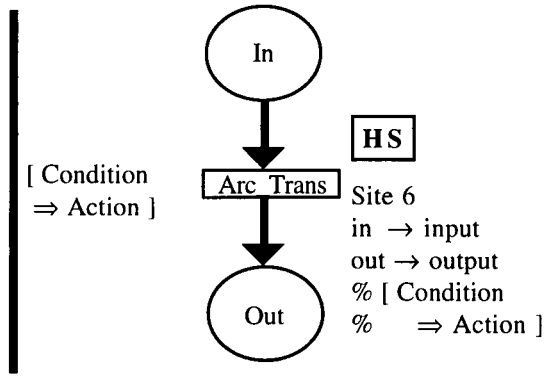


Figure 6.6.7.1 The Arc Petri Net SuperPage Model

Figure 6.6.7.2 depicts the sub page of the coloured Petri net hierarchy model of the arc construct. In this model the informal condition action statements that are normally associated with an arc are mapped down into formal statements. The statements are then scripted into the modelling language CPN ML and attached to the sub page model.

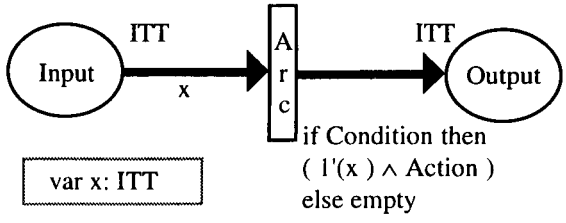


Figure 6.6.7.2 Site 6 The SubPage Model of the Arc Construct

Chapter 7 *The Man in the Van Case Study*

7.1 Introduction

In this chapter the first of two case studies will be presented. This case study is called *The Man in the Van Case Study*, and is taken from a regional Electricity Board. The electricity board is responsible for supplying electricity and electrical services to a large region, covering several large suburban and rural areas. The validation of the hypothesis (See Chapter 1) against this case study will take the following two forms:

- An enterprise model of the current system will be developed using the ideas and notations presented in this thesis. This model will then be compared and contrasted with a model of the current system that was developed using various business analysis techniques (Eason, 1995; Eason, 1985; Eason, 1987).

- A enterprise model of the system as proposed will then be developed and this will then be compared with the enterprise model of the current system in an attempt to ascertain whether the models contain any unexpected and significant results or differing views.

7.2 The Man in the Van Case Study

7.2.1 Introduction to the Case Study

Electricians in the Area Board provide a wide range of services on customer premises, such as repairing appliances and restoring supply. This can be a costly and wasteful process with electricians in vans inefficiently deployed, clerical staff unable

to process customer requests quickly, and long delays in communication. One problem is that the communication to arrange visits to premises for emergency work is undertaken by radio. This can often be sub-optimal, e.g. when the electrician is not in the van, the radio operator has to stack calls and keep trying. This case study is concerned with the installation of mobile data terminals in vans which can receive and store calls. The purpose is to examine the organisational implications of setting up a technical system of this kind.

The specification of the system as it currently exists is derived from the application of standard business analysis practices to the problem and is taken from (Eason, 1988; Eason, 1995; Eason, 1985).

7.2.2 The System as it Exists

Customer Service in the Area Board is organised as depicted in Figure 7.2.2.1. Figure 7.2.2.1 is an organisational structure chart (Tosi,Rizzo, & Carroll, 1994) of the electricity board from the Area Board down. The Senior Customer Service Engineer is directly responsible to the Area Commercial Manager for a Customer Service Unit based at Area Offices. The Customer Service Unit consists of a customer services unit manager, a team of clerks and a radio operator controlling communications with the fleet of service vans, each service van belonging to an electrician.

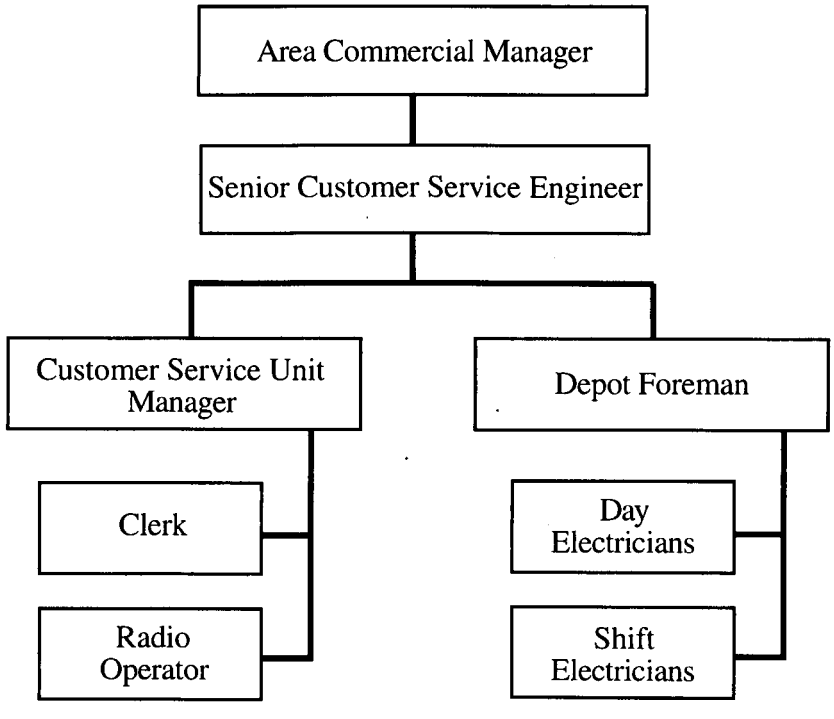


Figure 7.2.2.1 Customer Service Organisation Chart

There are four depots, each with a foreman supervising the work of a team of electricians who are of two types. Day electricians mostly undertake specialist activities requiring specialist knowledge, e.g. fixing and maintaining washing machines, whilst the shift electricians are responsible for the statutory duties of the Board, e.g. attending to main fuses on a property to restore supply. The underlying flow of work found in the case study is depicted in Figure 7.2.2.2

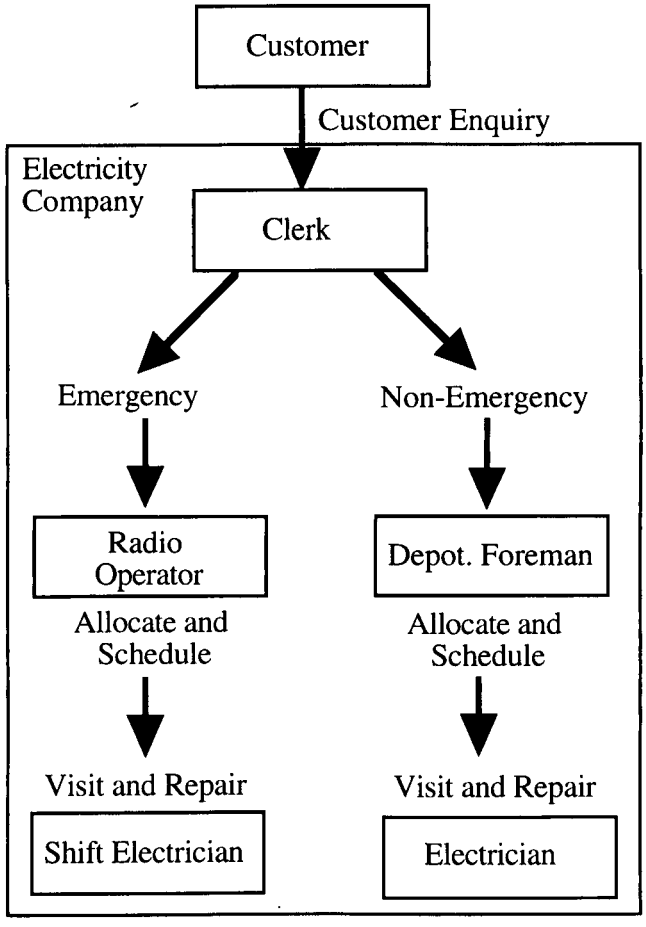


Figure 7.2.2.2 Current Work Flow in the Customer Service Unit

The work flow begins with a request from a customer. Each customer's request is routed to a clerk in the Customer Service Unit (CSU). At the CSU the request is received and logged by a clerk. The clerk decides if the requests are emergency calls (off-supply calls, dangerous situations, etc.) or non emergency calls (to repair cookers, central heating systems etc.). The emergency calls are given priority and should be dealt with straight away, as opposed to the non emergency calls which simply require a visit in three days. The electricity company has a series of depots to cover each part of the geographical areas of the company and a team of electricians operate from each depot. For both non-emergency calls and emergency calls the clerk raises a Work Instruction Sheet (WS). The WS is logged by the clerk on an on-line computer system which can be accessed at any time by the foreman. For non-emergency calls the clerk

allocates it to an electrician at the appropriate depot. The foreman checks the content of the WSs, raises the appropriate parts from the store, discusses jobs with electricians and may reallocate if necessary.

Emergency calls are passed to the radio operator who allocates them by radio primarily to the shift electricians. If the electrician is out of the van or the radio is busy, it may take a considerable time to make this contact. When the work is done, the electrician returns to the depot. and completes a job sheet that the foreman at the depot checks. The foreman then sends the WS by courier to the Customer Service Unit for de-programming, e.g. billing, arranging the next visit, etc. Once the work is completed by the electrician a work sheet is created and completed. This is then passed to the foreman who decides if the task was performed correctly based on the WS.

7.2.3 The System as Proposed

The electricity company plans to improve customer service because the present system has several faults. The process of dealing with non-emergency work takes a minimum of one day and if a job is incorrectly defined then it could take longer. The emergency route could be much faster but the radio contact depended on the electrician being in the van and able to receive the call. If the electrician was on the customer's property undertaking a repair, the radio link was lost and the call had to be stacked for later.

The management of the Board plans to introduce a computer-based mobile communications system with a mobile data terminal in each van. The terminal will be capable of receiving and storing up to 25 complete job messages. These can be displayed on a screen or printed at the electrician's discretion. The messages will be transmitted from a computer terminal operated by the radio operator in the Customer Service Unit. The system as proposed will enable the radio operator to send messages to a van giving the address of the customer and the job that had to be undertaken. Once the terminal in the van has acknowledged receipt of the message then the job is considered to have been allocated. The great benefit of this system is that the message could be transmitted while the electrician was not even in the van. A schematic of this communication system is provided in Figure 7.2.3.1. In this communication system emergency jobs are automatically allocated and scheduled by the radio operator. Non-Emergency jobs are given to the foreman to be scheduled and allocated. The foreman then informs the radio operator of where the job has been allocated. The process of communicating with the emergency and non-emergency electricians to inform them of their next job is now done by the radio operator, and from this point on the jobs are

treated as the electrician's responsibility. The management are looking for the following kinds of benefit from the system: faster response to emergencies, prompter response to non-emergencies, better customer services, better utilisation of electrician resources and more efficient use of office staff.

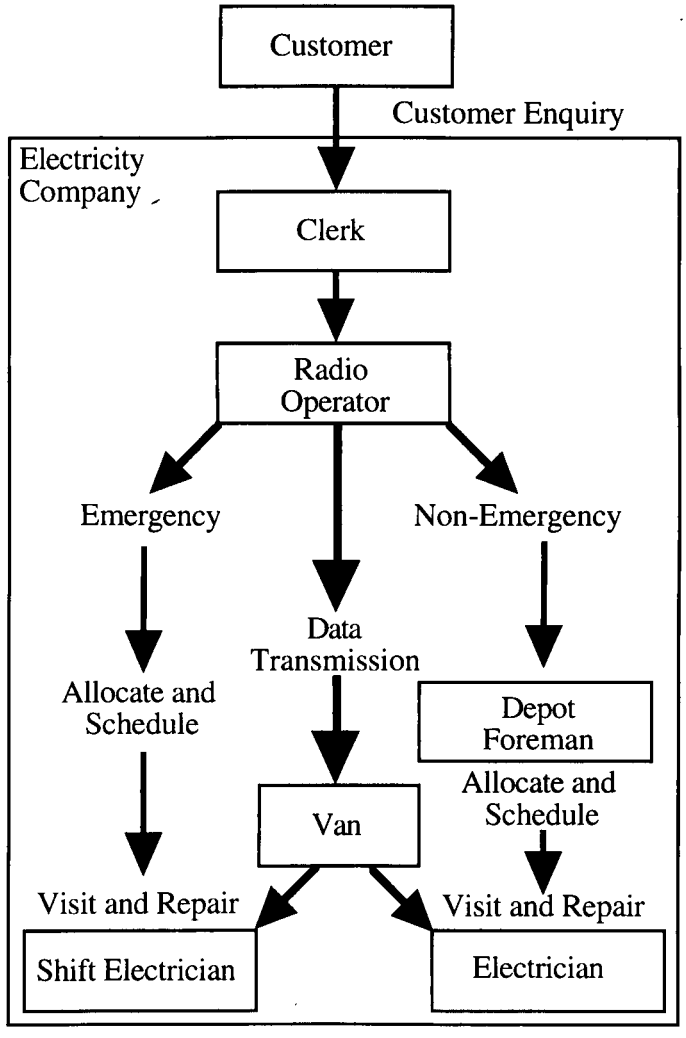


Figure 7.2.3.1 Proposed Work Flow in the Customer Service Unit

7.3 Models of the System as it Currently Exists

7.3.1 Structural Analysis of the Problem

The aim of using this case study is to demonstrate how the various notations presented previously can be used to elicit, validate and reason about, and thus derive, various requirements. In Figure 7.3.1.1 the two main agent entities that are involved in the case study are depicted. The agent entity on the left is the house–holder and the agent entity on the right is the Electricity Board. This diagram shows us that the house–holder has a *consumer–supplier* structural relationship with the Electricity

Board, this being represented in real life as the service contract that exists between the two agent entities.

The functional relationships show us the nature of a structural relationship in that the house–holder can notify the Electricity Board of a problem and the Electricity Board then has a responsibility to fix the problem. These two functions are illustrated in the diagram by the functional relationships *problem notifier–problem reception* and *problem owner–problem repair.* The diagram thus embodies the policy that the problem repair is effected by the supplier and the problem notification is effected by the consumer.

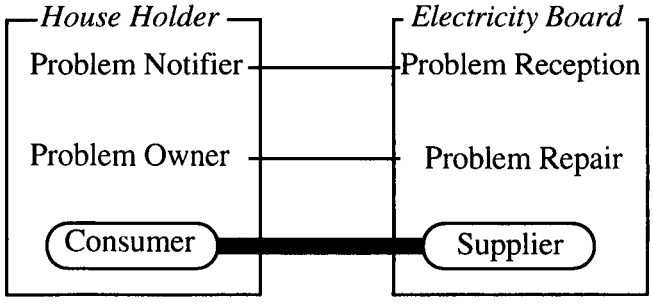


Figure 7.3.1.1 A Simple Beginning

When constructing a set of enterprise (role relation) diagrams it is important to keep in mind the types of questions that the diagrams attempt to answer. In this case study the questions that we are attempting to answer are concerned with the reorganisation of the Electricity Board, as it is planning to reorganize and restructure its operations. We proceed by decomposing the Electricity Board into its sub-components as depicted in Figure 7.3.1.2.

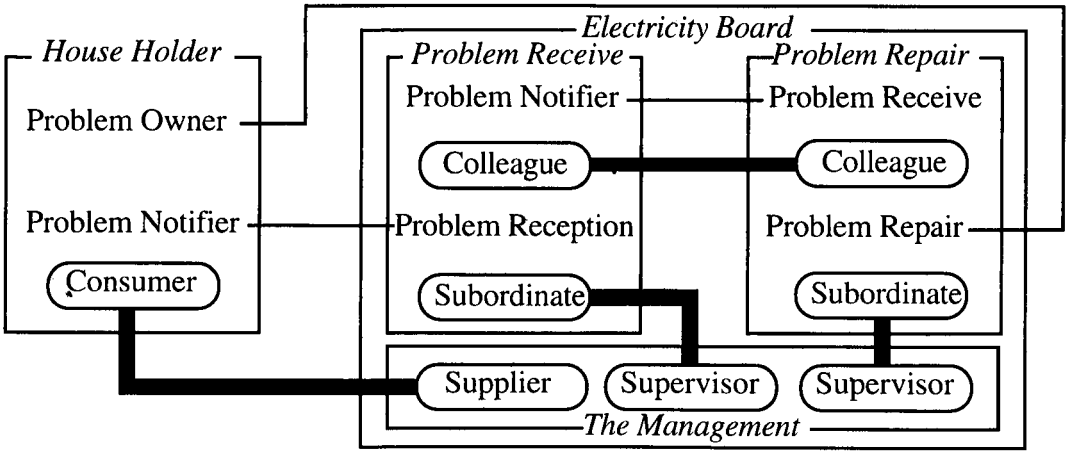


Figure 7.3.1.2 Decomposing the Organisation

By decomposing the Electricity Board into its sub-agencies as depicted in Figure 7.3.1.2 we can begin to explore and draw out the implications of how the organisation is structured. We can already begin to use these diagrams to elicit requirements by asking such questions as "what is the communication medium through which the functional relationships flow?", and "what are the responsibilities that reside within the structural relationships?"

With Figure 7.3.1.2 we can already see several important requirements emerging. For example the supplier relation now resides with an agent entity called *the management*. This agent entity has the responsibility to supervise the work that is carried out as a result of the *consumer–supplier* relationship which exists between the Electricity Board and the house–holder. We can also see various legal implications emerging as the legal responsibility for the actions performed by the organisation now resides with *the management* agent entity. If the house–holder were to take the Electricity Board to civil court and they were to be found liable and ordered to pay recompense then this order would fall upon the management agent entity.

Figure 7.3.1.3 shows the decomposition of the *problem repair* agent entity into three sub agent entities, *Foreman*, *Emergency–Man* and *Ordinary–Man*. The colleague relationship has been expanded to link into all the sub–agent entities and the supervisor–subordinate structural relationship has been connected to the *Foreman* agency thus illustrating the chain of command.

Figure 7.3.1.3 depicts the sub-agencies and the relationships that exist not only with each other but also with the outside world for the problem repair agent entity. This diagram clearly shows the three sub agent entities *Foreman*, *Emergency–Man* and *Ordinary–Man* along with how the structural and functional relationships have been decomposed. This diagram illustrates an important point that should be noticed. This is that the colleague structural relationship has been decomposed and bound into each of the problem repair sub agent entities. The supervisor–subordinate relationship that exists between the management and the problem repair agent entity has been bound into the foreman agent entity. This represents the organisational management structure that exists within the Electricity Board.

The *problem notify–problem repair* functional relationship has also been decomposed down into *emergency problem reception* and *ordinary problem reception* for the emergency man and ordinary man agent entities respectively. This represents the classification and allocation of problems that the problem reception agent entity performs with the emergency man and ordinary man agent entity. It also represents the delegation of the classification and allocation of the problems to the foreman, but

this can only occur under certain conditions. Thus we may ask the question, under what conditions can the problem reception agent entity delegate the classification and allocation of the problem to the foreman ?

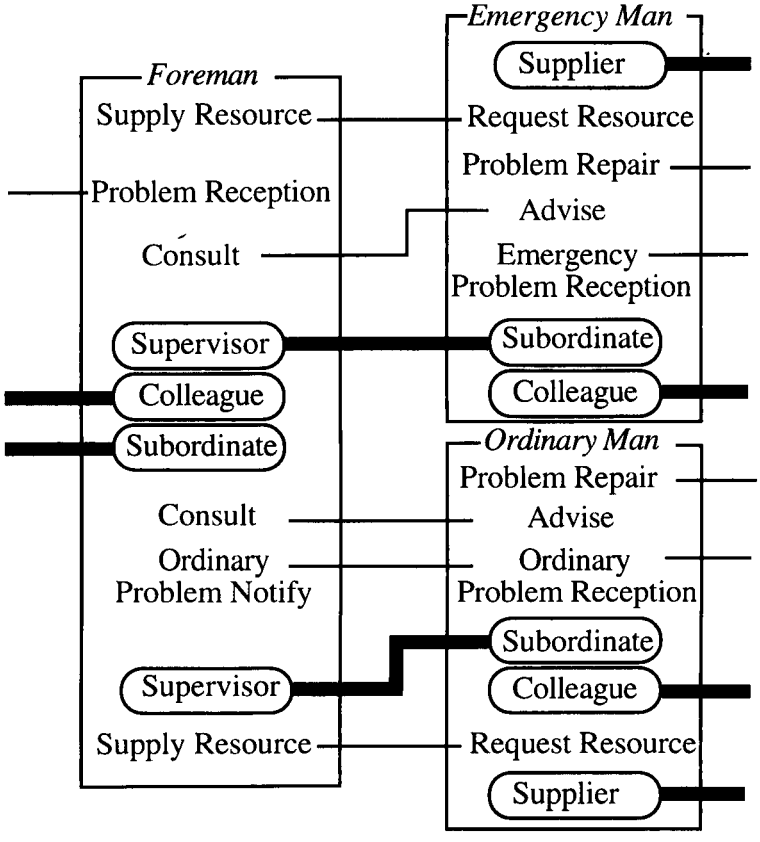


Figure 7.3.1.3 Decomposing the Problem Repair Agency

Enterprise diagrams like those depicted in Figure 7.3.1.3 allow us to explore and define where the organisational boundaries lie. For example, when constructing the enterprise diagram shown in Figure 7.3.1.3 we could define the emergency–man and the ordinary–man to lie outside the organisation. This would mean that we had defined the *emergency–man* and the *ordinary–man* to be subcontractors to the *electricity–board*. Having defined our organisational boundaries we can then examine the conversations to see what effect sub-contracting will have to a) the flow and fulfillment of the electricity-boards responsibilities, and b) the utilisation of organisational resources. Given that the electricity–board fulfills its responsibilities through the actions of the *emergency–man* and the *ordinary–man*, the *electricity–board* would be ill advised to sub-contract that work out to another third party without prior careful examination of the flow and manipulation of responsibilities.

7.3.2 Interaction Analyses of the Problem

In this case study the communications occur in the first instance between the consumer (the problem owner) and the Electricity Board itself (the problem repairer). Then, through the Board's own internal channels of communication, the nature of the problem is received by the electrician, whose task it is to repair it. Supervision occurs between the foreman and both the emergency electricians and the ordinary electricians.

In the Electricity Board System as it stands as stated in section 7.2, a clerk receives the customer's call at the Customer Service Unit (CSU) and makes a decision as to whether it is a emergency or not. It is classified either as a non-emergency, requiring a visit within three days (type A), or as an emergency requiring immediate action (type B). If the customer's premises have to be visited, the problem is recorded as a task for the electricians.

If the task is of type A, the Clerk raises a Work Instruction Sheet and allocates it to a day electrician at the appropriate depot. The foreman has access to the Work Instruction Sheet via a computer. When the foreman has checked it, the foreman may then discuss the job with the day electrician and raise the necessary parts from the store. Under certain conditions the foreman may even reallocate the job. If the task is of type B, the radio operator allocates the job to a shift electrician at an appropriate depot via radio. When work is completed (both types A and B) the electricians complete job-sheets, which are checked by the foreman at the depot, and are then sent to the Customer Service Unit for billing and such like.

At this stage we are concerned only with the conversations arising out of non-emergency situations (type A). From the point of view of analysing them by interaction analysis, their dynamics make them more interesting and therefore they are likely to prove more insightful. Since the structural relation between the two individuals is one of supervision, we want to be able to assess whether the graphs are able to represent this adequately.

The first point to note about Figure 7.3.2.1 is that it depicts four of the basic building blocks of an interaction diagram. Starting at the top of the diagram and working down we note that the beginning of the interaction is depicted with a start/finish indicator. The next component in the interaction diagram is a speech act, and this object is depicted as a rectangle. The speech act is composed of three components, the first of which is the agent entity that is making the utterance and the second of which is the type of utterance that is being made. The third and final

component is the enterprise object that is being accessed in some way by the utterance. Since the expression of the enterprise object's manipulation is an optional component of the speech act it is denoted by a thick black line at the left hand side of the component.

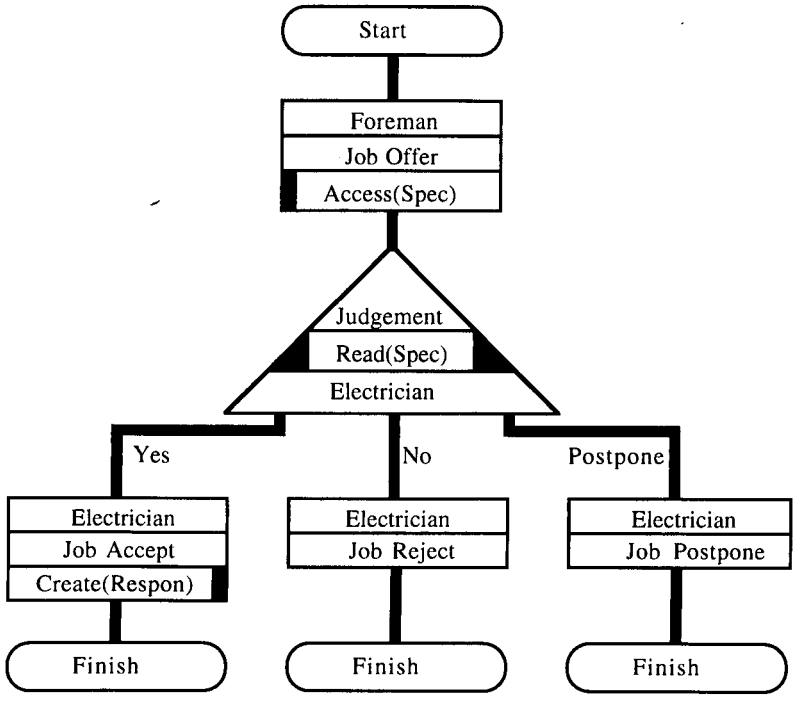


Figure 7.3.2.1 The Offer Stage

The largest of the basic building blocks that is shown in Figure 7.3.2.1 is called a decision act. This act is also composed of three components, the first of which expresses the type of decision that is being made. The second is an optional component to the act and expresses the conditions necessary for an agent entity to make the decision. Since this is an optional part of the act it is denoted by two thick black shaded areas on either side of the condition. The third component of the act is the agent entity that is engaged in making the decision. The fourth and final basic building blocks that are shown in Figure 7.3.2.1 are used to denote the termination of the interaction and are called finish indicators.

Figure 7.3.2.1 shows us the stage in the conversation at which the foreman offers the job to the electrician. This diagram serves to point out the three possible ways in which the electrician may respond to the job offer speech act made by the foreman. The electrician may either postpone the job, reject the job, or accept the job. In performing the latter speech act the electrician is entering into a responsibility called *Respon* with the foreman, that responsibility being to perform the job specified by the foreman.

The speech act on the far left of Figure 7.3.2.2 is noteworthy in that it depicts the creation of an enterprise attribute. Since the expression of the manipulation of the enterprise attribute is an optional component of a speech act, it is denoted by a thick black line at the left hand side of the component. The building block at the centre of the interaction diagram that is shown in Figure 7.3.2.2 is called an instrumental act. This act allows for the expression of the nature of the task that an agent entity is required to perform. The optional part of the act is shown in the diagram as having two black shaded lines on either side of the component. This optional component allows for the representation and examination of the enterprise objects and attributes required for, or altered by, the performance of the instrumental act.

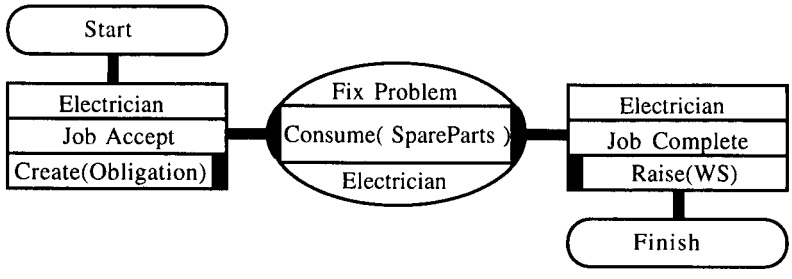


Figure 7.3.2.2 Performing the Task

Figure 7.3.2.2 represents how the interaction between the foreman and the electrician further develops. In this interaction diagram the electrician is shown accepting the job offer made by the foreman and thus entering into an obligation with the foreman. The electrician then performs the task specified by the foreman and in so doing consumes some spare parts.

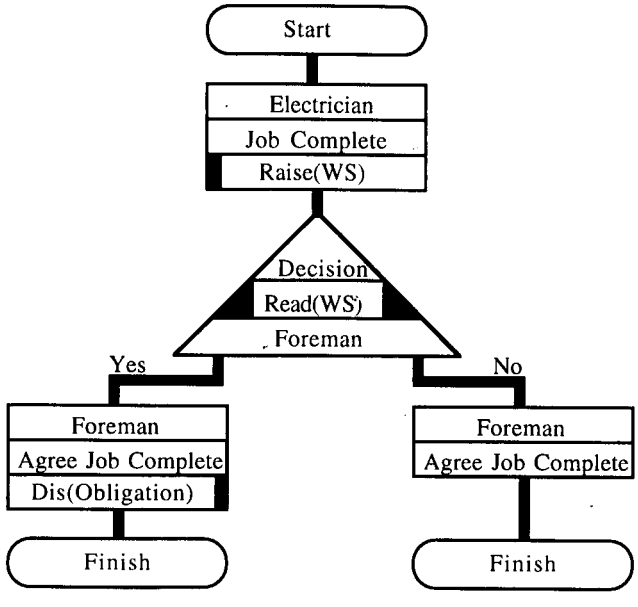


Figure 7.3.2.3 Discharging the Obligation

Upon completion of the task the electrician reports back to the foreman and raises a work sheet (WS). Once the electrician has raised a work sheet (WS) the foreman then reads the WS and decides if the electrician was successful in the completion of the task. The decision of the foreman has two possible outputs, either yes or no (See Figure 7.3.2.3). If the foreman chooses the former, then he agrees to the electrician's appraisal of the situation and thus the electrician's obligation that was created earlier is discharged. In the above interaction diagram we have shown that it is the act of the foreman that discharges the obligation which was held by the foreman. If the foreman chooses the latter, then the job sheet must be amended to the satisfaction of both of them before being passed to the CSU.

When the job is allocated to the electrician and the electrician either rejects it, or postpones it, then the foreman may request an explanation from the electrician. In the interaction model that has been presented in this thesis the conversations are shown as stopping. In reality it would be the problem solvers and problem owners that would dictate when the interaction modelling should stop.

The beginning of the process of scheduling the electrician is shown in Figure 7.3.2.4. This process starts with the clerk making a job offer to the electrician and filling out a work instruction sheet. The filling out of the work instruction sheet that is performed by the clerk is expressed in the enterprise component of the speech act as *Write(Spec)*.

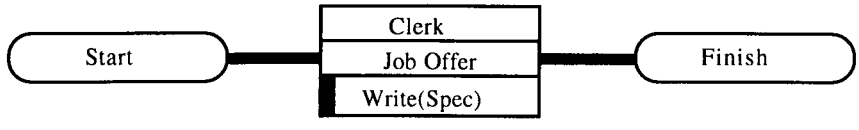


Figure 7.3.2.4 Beginning the Scheduling

Once the job offer has been made by the clerk the electrician then has to make a decision similar to the decision act shown in Figure 7.3.2.1. As with the decision act that is depicted in Figure 7.3.2.1 the decision that the electrician is required to make has three possible outcomes. Having made a decision the electrician then has to generate the appropriate response.

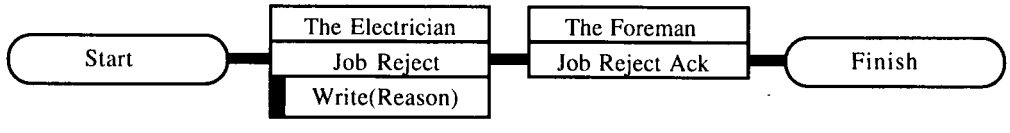


Figure 7.3.2.5 The Job Rejection.

Figure 7.3.2.5 shows the electrician making a job rejection utterance followed by the foreman making a job rejection acknowledge utterance. When we use

interaction diagrams to examine the scheduling of the electrician we notice that as with Figure 7.3.2.5 after the initial job offer made by the clerk all interaction takes place between the electrician and the foreman. So, understanding the relationships that can exist between the foreman and the electrician is vital in comprehending how the organisation discharges its obligations and responsibilities to the house–holder.

7.3.3 Analysis of Results

In this section I shall compare and contrast the enterprise model of the current system with a model of the current system that was developed using various business analysis techniques (Eason, 1995; Eason, 1985; Eason, 1987). The purpose of this analysis is to ascertain whether or not the modelling notations and techniques developed in this thesis are better than the standard business analysis techniques

In the business analysis presented in (Eason, 1995; Eason, 1985; Eason, 1987) and in Section 7.2 the organisational structure is depicted by an organisational structure chart (Tosi et al., 1994). Within most business analysis techniques organisational structure charts are used to identify and validate potential stakeholders and to define organisational boundaries. Organisational structure charts allow us only to depict supervisor–subordinate relationships (See Figure 7.2.2.1). These types of charts only allow you to explore and analyse the organisation from an *is–supervised–by* perspective. Thus the organisational structure chart depicted in Figure 7.2.2.1 tells us that the clerk and the radio operator are supervised by the Customer Service Unit manager, and that the Customer Service Unit manager is supervised by the senior customer service engineer.

The enterprise diagrams developed in this thesis and used in this case study allow us to depict not only supervisor–subordinate relationships, but also client–server and peer–peer relationships. Organisational structure charts do not allow us to depict how responsibilities flow through an organisation, or bind to individuals. The enterprise modelling language makes use of the concept of responsibility to define the supervisor–subordinate, client–server and peer–peer relationships. It uses responsibility as a mechanism to define a) many different types of relationships, b) organisational boundaries, c) the organisational resources required to support the relationships and d) the set of interactions required to mediate the responsibilities. As a result we have found the diagrams like Figure 7.3.1.2 to be of more use than structure charts when conducting stakeholder analysis and defining organisational boudndaries. Through the use of responsibility modelling these diagrams allow us to examine the organisation from a wider perspective, and thus to capture more stakeholders than traditional stakeholder analysis. For example, when analysing the

responsibility that exists between the electricity–board and the house–holder, we can ask the question "Who adjudicates this responsibility?" Having identified another agency we could then ask "Does this agency need any access rights over the information contained in the information system?" If the answer to this question is yes then we have identified another stakeholder. In addition, we can also ask "Does the adjudicating agency form part of the organisational system, and if not then where does the organisational boundary lie?"

For example, Figure 7.3.1.2 tells us that the house holder has a consumer–supplier relationship with the Electricity Board. The enterprise modelling language defines what the consumer–supplier relationship means. We define what responsibilities and resources exist within the relationship when we instantiate the relationship within a model. The functional relationships that exist in the model between the house holder and the Electricity Board tells us how the relationship may be invoked. Figure 7.3.1.3 tells us that the foreman is supervising the work of the ordinary electrician by means of the supervisor–subordinate relationship. The enterprise modelling language defines what the supervisor–subordinate relationship means. We define what responsibilities and resources exist within the relationship when we instantiate the relationship within a model. The functional relationships that exist in the model between the foreman and the ordinary electrician tell us how the relationship may be invoked, and how the responsibility within that relationship is mediated. The enterprise modelling language provides a much richer vocabulary (compared to structure charts) through which we can express more accurately the structure of the organisation.

The work flow model of the system as exists (See Figure 7.2.2.2) shows the one way flow of information through the organisation. It views the flow of information through the organisation as a directed graph. For example, Figure 7.2.2.2 tells us that information flows from the clerk to the radio operator. It does not however tell us:

- anything about the mediation process which results in the job being allocated, performed and evaluated.
- if any information flows back from the radio operator to the clerk, or how the radio operator and the clerk communicated.
- what organisational resources were used to store the information, or how those organisational resources were accessed.
- when, where, and by whom decisions are being made.

- if as a result of the flow of information between the radio operator and the clerk any of the organisation's responsibilities or obligations were fulfilled or discharged.

The model of organisational behaviour developed in Chapter 6 and used to model the communication between the radio operator and the clerk (See Figure 7.3.2.4 and Figure 7.3.2.5), and the foreman and the electrician (See Figure 7.3.2.1, Figure 7.3.2.2 and Figure 7.3.2.3), addresses all of the above points. The model of organisational behaviour used in this thesis views communication as a directed graph, and consequently it allows us to capture everything that the work flow model presented in Figure 7.2.2.2 allows us to capture.

This model of organisational behaviour uses the concept of speech acts (Searle, 1969) to define and analyse the communication that can exist between two or more parties. Speech acts allow us to define and analyse the nature and type of the communication. From the application of speech acts to the modelling of communication we can define the information that flows between the various parties, and the resources and access rights that are used to encapsulate, define and manipulate that information. In addition, in Figure 7.3.2.1 and Figure 7.3.2.3 we can see how we have extended the speech act model to allow us to capture and reason about when, where, how and by whom decisions are made. The result of this modelling is that we can see what social values (responsibilities) are being mediated and how they are being mediated.

For example, in Figure 7.2.2.2 we can clearly see information flowing from the clerk to the foreman, and from the foreman to the electrician. However, what we cannot see is the mediation process that goes on between the foreman and the electrician, and by which a job can be re-allocated to another electrician. This mediation process is clearly depicted in Figure 7.3.2.1. In Figure 7.3.2.1 we can see the foreman making an offer to the electrician and the electrician making a decision as to whether or not to accept the offer. Consequently we can see the information flowing from the foreman to the electrician. What Figure 7.3.2.1 gives us that is different from Figure 7.2.2.2 is: a) the ability to see information flowing back from the electrician to the foreman, i.e the electrician saying no to the offer and the job being re-allocated, and b) the ability to see when, where and by whom decisions within the organisation are being made.

Finally, the speech act model also allows us to define and analyse when, where, how and by whom responsibilities or obligations are created, manipulated and destroyed. This analysis is vital to the success of any organisational re-engineering

activity. The procurement and development of any organisational information system will definately involve organisational re-engineering. Two things have influenced the organisation more than anything else; the coffee machine and the computer. The reason for this is that they both mediate social relationships (Ehn, 1989). It is therefore vital that when re-engineering the organisation we take account of how these social relationships are affected (Dahlbom & Mathiassen, 1994). If we fail to take into account how social relationships are mediated in the old and new systems then the system is likely to fail and the resulting failure could cost the organisation millions or even a person their life (South West Thames Regional Health Authority, 1993; Wiener, 1993). In modelling and analysing how responsibilities flow through an organisation, and are mediated by the organisation, we can begin to build up a picture of what social relationships, and social values, are encapsulated within the organisation. Thus when re-engineering the organisation the notation developed in this thesis will provide the problem owners and problem solvers with a greater chance of success than that used in Figure 7.2.2.2 and Figure 7.2.3.1 to analyse the organisation.

7.4 Models of the System as Proposed

7.4.1 Structural Analysis of the Problem

Figure 7.4.1.1 shows the new organisational model that is being proposed for the new system. This diagram uses the same notation as those presented in Section 7.3.1.

In Figure 7.4.1.1 we can see that an answering machine agency called *The Machine* has been added to the *Ordinary Electrician* agency (The Man in the Van). The radio operator still has a colleague–colleague structural relationship with the electrician, foreman and clerk. However the nature of the interactions between the electrician and the foreman has changed. The radio operator also has a new consumer–supplier structural relationship with the machine. As a result when the radio operator allocates a job to a particular electrician a message is sent to the machine. This message contains the description of the problem and the address of the problem owner. Once a message has been sent to the machine then the responsibility for the completion of the work is said to have been allocated to the electrician. The radio operator still has a colleague–colleague structural relationship with the clerk and the electrician and a supervisor–subordinate relationship with the Customer Service Unit manager.

The electrician still has a colleague–colleague structural relationship with the radio operator, a supervisor–subordinate relationship with the foreman, and a supplier–consumer relationship with the problem owner. The nature of the interactions between the radio operator and the foreman have changed. The emergency man now has a consumer–supplier structural relationship with the machine. The relationship defines that the emergency man can retrieve and store messages from the radio operator. The supervisor–subordinate structural relationship with the foreman tells us that while the foreman is still supervising the work of the electrician there is no easy way for the electrician to get in touch with the foreman.

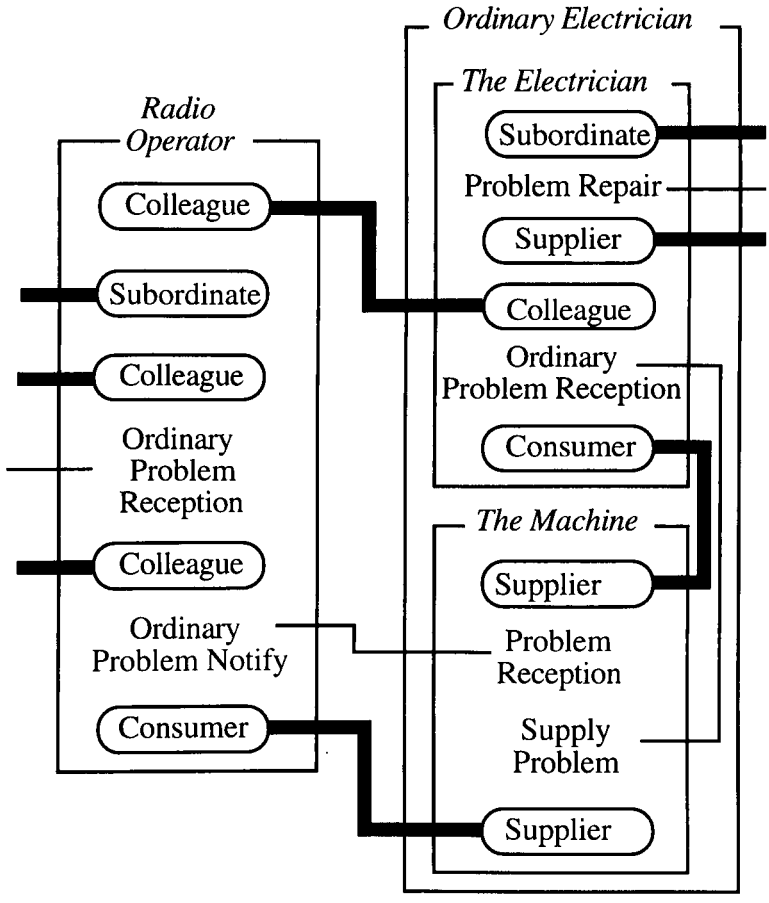


Figure 7.4.1.1 The New Repair Agency

7.4.2 Interaction Analyses of the Problem

In Section 7.2.3 the system as proposed is stated. In the system as proposed all communication to the emergency and non-emergency electricians is now routed through the answering machines inside the electrician's vans, by way of the radio operator. The same notation will be used in this section to model interaction as was used in Section 7.3.2. Figure 7.3.2.1 and Figure 7.3.2.2 depicts how a job under the system as proposed will be allocated to an electrician.

The interaction depicted in Figure 7.4.2.1 starts with the radio operator contacting the machine in the electrician's van and defining the next job that the electrician is required to perform. There are two points that should be noted about the interaction performed by the radio operator. The first is that the type of utterance made by the radio operator is not a request, but a command. Thus the machine has no choice but to accept and record the job. The only grounds on which the machine can reject a job is if the machine is faulty. The second is that while making the utterance the radio operator is reading from the work instruction sheet (WS) which defines the job that the electrician is required to perform.

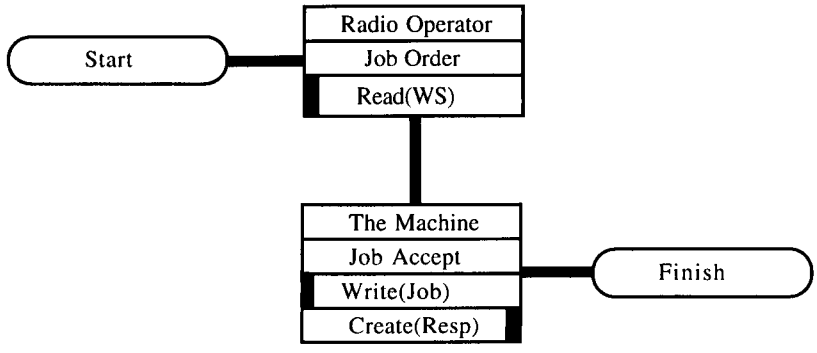


Figure 7.4.2.1 Interaction between the Radio–Operator and the Machine

The interactions depicted in Figure 7.4.2.1 continue with the machine in the van acknowledging reception and storage of the message. In sending this acknowledgement back to the radio operator is machine is creating a responsibility which binds to the electrician. Thus after this utterance by the machine the electrician is now responsible for performing the job specified in the message. In addition, sending the acknowledgement back to the radio operator also signifies that the machine has successfully recorded the job description for the electrician.

The interaction depicted in Figure 7.4.2.2 starts with the electrician commanding the answering machine in the van to display the next job. We should note that this utterance made by the electrician is a command and not a request. Consequently the machine has to make a deterministic exclusive–or decision. If the machine knows about a job then it has to display it to the electrician, and if not then it has to report saying that no more messages are available. The types of utterances made by the machine for both the yes and no answers to the command to display the next job by the electrician are propositions. In both cases the machine is asserting a fact; either there are no more jobs for the electrician to perform, or there is a another job for the electrician to perform.

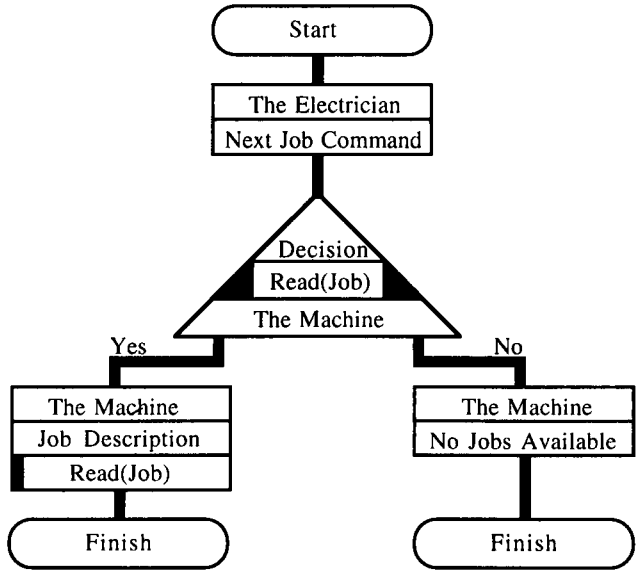


Figure 7.4.2.2 Interaction between the Electrician and the Machine

7.4.3 Analysis of Results

In this section I shall compare and contrast the structural and interaction models of the system as it currently exists and of the system as it is proposed. The purpose of this analysis is to attempt to ascertain whether or not the models of the man in the van developed in this thesis contain any unexpected and significant results.

Comparing the structural models of the system as it currently exists (See Figure 7.3.1.1, Figure 7.3.12 and Figure 7.3.1.3) and the system as proposed (See Figure 7.4.1.1), we can observe some structural changes. The first point is that the nature of the colleague–colleague relationship between the electricians and the radio-operator has changed. In the system as proposed the duties and responsibilities of the radio operator with regard to the electricians have been expanded. In the current system only the emergency electrician has any interaction with the radio–operator and that is when the radio–operator informs the electrician what the next job is. In the current system if the emergency electricians does not like the job that has been allocated to them, they can communicate via the radio with the foreman and attempt to get the job allocated to another electrician.

In addition, the current system is quite different from the system as proposed when we compare the roles played by the foreman. In the current system the foreman supervises the work of the electricians through constant communication with them. If the electrician has any problems with a job then the electrician can get in touch with the foreman and seek advice. If the electrician is not able to perform the job then the electrician could get in touch with the foreman and seek to get the job reallocated. In

the system as proposed the foreman no longer has any direct communication with the electricians while they are out on a job.

In the structural models (See Figure 7.3.1.1, Figure 7.3.1.2 and Figure 7.3.1.3) of the system as it currently exists we can see the types of interactions that exist between the foreman and the electricians. In Figure 7.3.1.3 we can see that the foreman has the ability to reallocate a job through the consult–advise functional relationship for both the emergency, and non–emergency, electricians. In Figure 7.3.1.3 we can also see that the foreman has the the ability to supply the electricians with any resources that they need in order to fulfill their responsibilities and fix the problems. In the structural models (See Figure 7.4.1.1) of the system as proposed we can see that there is now no interaction between the foreman and the electrician. In fact 7.4.1.1 tells us that the only interaction that the electricians have is with the machines in their vans that are used to store messages. In the system as proposed the electricians no longer communicate with humans, only with a machine.

Comparing the interaction models of the system as it currently exists (See Figure 7.3.2.1, Figure 7.3.2.2, Figure 7.3.2.3, Figure 7.3.2.4 and Figure 7.3.1.5) and the system as proposed (See Figure 7.4.2.1 and Figure 7.4.2.2), we can observe some changes in the nature and type of the interactions. The first observation is perhaps the most important, and that is that by comparing the interaction models on the whole we notice that the communication for the system as proposed is all one way. In the current system the electrician has the ability to have a two way communication with both the radio operator and the foreman by means of the radio contained in the vans. This two way communication is vital in order for the foreman to supervise the work of the electrician. In the system as proposed the radios in the vans have been replaced by data terminals. These terminals only allow for the storing and forwarding of messages from the radio operator to the electrician. Consequently in the system as proposed there is no way for the electrician to communicate with either the foreman or the radio operator.

When examining the interactions between the foreman and the electrician that exist in the current system and are depicted in Figure 7.3.2.1, we can observe that the electrician only enters into a responsibility with the foreman after a negotiation process. In this negotiation process the electrician has the opportunity to say no to the job being offered and thus refuse the responsibility. While examining the interactions between the foreman and the electrician that exist in the proposed system and as depicted in Figure 7.4.2.1 and Figure 7.4.3.2 we can see that the responsibility for an electrician to perform a job is created by the machine and not the electrician. The responsibility is created when the machine acknowledges the receipt and storage of a

message from the radio operator. In examining Figure 7.4.2.1 and Figure 7.4.3.2 we can observe that the electrician has no way to refuse.

7.5 Concluding Remarks on the Case Study

The purpose of this case study was to validate the hypothesis contained in Chapter 1 of this thesis. The validation of the hypothesis takes the following two forms:

- An enterprise model of the current system will be developed using the ideas and notations presented in this thesis (See Section 7.3). This model will then be compared and contrasted with a model of the current system that was developed using various business analysis techniques (Eason, 1995; Eason, 1985; Eason, 1987).

- A enterprise model of the system as proposed will then be developed and this will then be compared with the enterprise model of the current system in an attempt to ascertain whether the models contain any unexpected and significant results or differing views.

The first form of the validation of the hypothesis is addressed in section 7.3. In this section an enterprise model of the current system is developed using the ideas and notations presented in this thesis. This model is then compared and contrasted with various business analysis techniques that have been applied to the problem (Eason, 1995; Eason, 1985; Eason, 1987). The business analysis techniques provided us with a very power oriented view of the system. The enterprise structural models (See Section 7.3.1) of the organisation used in this thesis take a wider, more diverse view of types of relationships that can exist. Consequently the structural models allow us to develop a more expressive picture of the organisation. The information flow models of the system as proposed and contained in (Eason, 1995; Eason, 1985; Eason, 1987) provide us with the ability to examine where information is flowing within the organisation. It does not provide us with the ability to capture any direction flow of information, or the interactions that led to the flow of information. The interaction models of the organisation (See Section 7.3.2) provide information flow models which allow us to capture the nature of the interactions and the nature and direction of the information flow.

The second form of the validation of the hypothesis is addressed in section 7.4. In this section an enterprise model of the system as proposed was developed and compared with a model of the current system (See section 7.3) in an attempt to

ascertain whether the models contain any unexpected and significant results or differing views. Through examining and comparing the structural and dynamic models of the organisation we can see how the roles of the radio operator, foreman and electrician have changed. In particular we can see how the duties of the radio operator have been expanded and the duties of the foreman have been reduced. The complete change in the nature of the communication between the electrician and the foreman and the electrician and the radio operator is something which is most unexpected and is very definitely significant. In the system as proposed the Electricity Board have refined the mediation process by which responsibilities are assigned. When the Electricity Board set out to introduce a new system their perception of the system was that it would speed up response time and make the organisation more efficient. They did not expect the re-organisation to redefine the work roles of the radio operator, foreman or electrician. As a result I conclude that the models of the system as it currently exists and the system as it is proposed contain unexpected and significant results.

Chapter 8 *The Jervis Street Hospital Case Study*

8.1 Introduction

In this chapter the second of two case studies will be presented. This case study is called The Jervis Street Hospital Case Study, and is taken from an inner city Irish hospital. This hospital is one of several that contains an A&E department in the wider metropolitan area of Dublin, Ireland and serves as a front line institution for the delivery of primary and secondary health care. The validation of the hypothesis (See Chapter 1) against this case study will take the following three forms:

- An information model of the current system will be developed and derived using the ideas and notations presented in this thesis. This model will then be compared and contrasted with a model of the current system that was developed using various business analysis techniques (IT-Uptake Project, 1986a; IT-Uptake Project, 1986b; Lorenzi,Riley,Ball, & Douglas, 1995) to see if the information model can: a) capture all the data contained in the various business analyses, and b) show up any contradictions and anomalies.

- A dynamic model of the current system will be developed and examined to see if the dynamic model can: a) capture all the data contained in the various business analyses, and b) show up any contradictions and anomalies.

- Both sets of models will then be presented back to the analysts and consultants that originally worked on the problem and their comments and suggestions elicited.

The references (IT-Uptake Project, 1986a; IT-Uptake Project, 1986b; Lorenzi et al., 1995) represent a record of the data which was gathered in the IT-Uptake

project. The data was gathered by means of observation of the work performed by the A&E department and interviews with all the A&E members of staff.

8.2 The Jervis Street Hospital Case Study

8.2.1 Introduction

This case study is taken from an accident and emergency department located in the inner city hospital of Jervis Street, Dublin. Various social, economic and political factors combine to influence this case study and make it extremely interesting. However for the purposes of this thesis, a narrowed down and more selected problem domain will be presented without generalization and analysed from only one perspective. It should be remembered that this problem forms part of a bigger problem, which in turn fits into the social, economic and political environment of the country.

The functions performed by an A&E department are basically two-fold. Its primary function, as its name implies, is to provide accident and emergency services whenever and to whomever is in need of them. This function has evolved into a second function, which actually forms the largest proportion of work performed by the department. This function is the provision of low grade medical services to the local population, where patients use the A&E department as an alternative to general practice. Two factors have contributed to this configuration of work within the department. The first is that the hospital does not charge for its services as opposed to some private practitioners. Secondly the A&E department at Jervis Street has traditionally had an open door policy towards all patients.

8.2.2 The Context of Health Services in Ireland

The health services in Ireland are administered by the Department of Health. This department has responsibility for the planning, co-ordination and administration of funding for health service provision. It develops the detailed allocation of funding to a number of regional Health Boards, and provides large, relatively undifferentiated, blocks of funding to the Health Boards for this purpose.

The Department of Health's principal functions are to plan and advise the Health Boards on the provision and funding of health services. In the context of IT, the Department of Health can, and does, advise the Health Boards on the types of functions suitable for IT. In addition they provide guidelines to health services in

general on what types of IT application systems may be implemented within health care contexts, and specify the potential vendors of these systems.

Within a health board, funding is applied to three types of programme: General Hospitals (e.g. Jervis St.), Special Hospitals and Community Care. By far the largest proportion of funding goes to the general hospital budget. Against this financial set–up the demands on services have been rising due to population growth and economic recession. This has placed more people in need of publicly funded health care services and has also tended to create an increased incidence of health problems in those sections of the population most vulnerable to economic recession. Much of the clientele that use the Jervis Street Accident and Emergency department services are drawn from inner city areas of Dublin.

8.2.3 Managerial Structure of the A&E Department

The staff structure is defined as follows and represents the managerial structuring of the accident and emergency department.

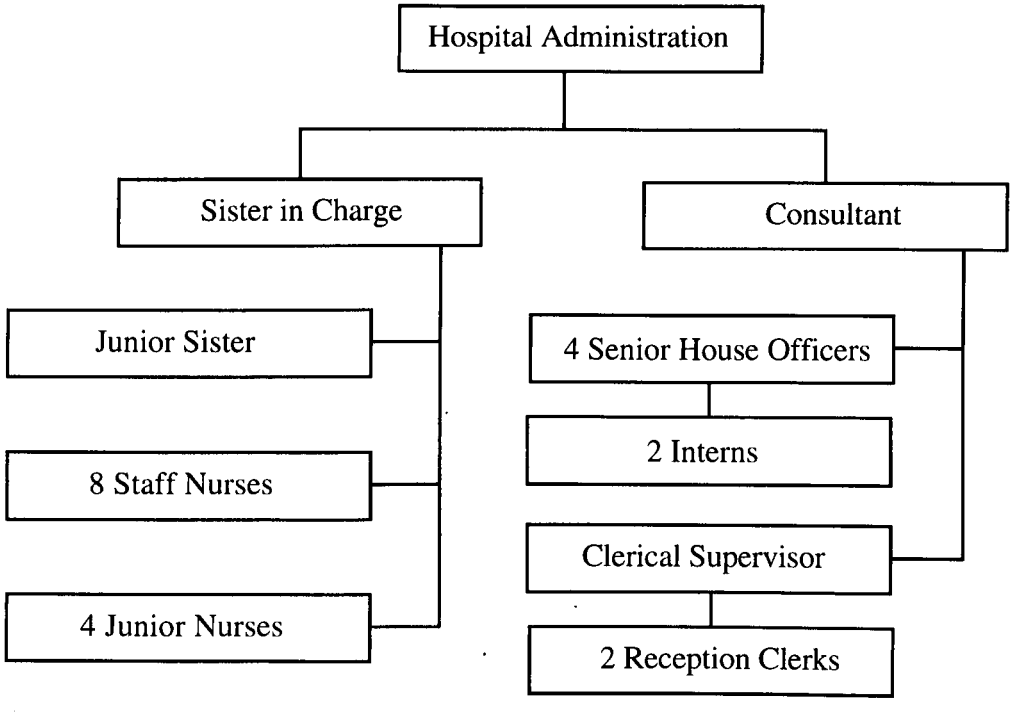


Figure 8.2.3.1 The Organisational Structure of the A&E

In Figure 8.2.3.1 we can see that the hospital administration is responsible for managing all aspects of the hospital and the various systems that make up the hospital. In addition, it is ultimately responsible for all actions performed by its staff. This means that it controls the purse strings for the various departments and that it defines the organisation's policies and mission statements.

The sister in charge is responsible for the behaviour of all the other sisters and nurses who work in the accident and emergency department, and reports directly to the hospital administration. The sister in charge is also responsible for the training of the student nurses.

The consultant is responsible for the day to day running of the accident and emergency department. Under the consultant there are 4 senior house officers who are attached to the unit for a 9 month period and 2 interns who are attached to the unit for a 3 month period. In addition, under the consultant there are three clerical staff; the clerical supervisor also functions as the consultant's personal secretary. The clerical staff for the whole hospital are subcontracted in from outside the hospital by the hospital's administration, and the administration pays for the clerical staff out of central hospital funds.

The A&E department is functionally related to a number of other departments within the hospital upon which it relies for services, e.g., pathology, radiology, pharmacy and the central administration. The A&E department internally consists of a small set of separate and distinct entities, e.g. senior nurse, consultant, junior doctor, junior nurse and receptionist.

Rather than describe the overall workings of the department, I will focus on one distinct function performed by the junior nurses within the department. In focusing on this distinct function I will attempt to derive from the junior nurse's perspective the information requirements of an IT system. The junior nurse function that I will focus upon is the administering of drugs to patients.

8.2.4 The Role of the Junior Nurse

For the purposes of this case study we will concentrate on the role of the junior nurse in the administering of drugs that have been prescribed to the patient. Under Irish law the administerer of a drug is responsible for the effect that the drug has on the patient, thus if the wrong drug is prescribed and administered then the administerer is held responsible under the law. This gives rise to the nurses having to perform, and being taught to perform, some of the functions normally associated with a doctor. When administering drugs to a patient, one nurse administers the drug while another nurse observes to make sure that the correct procedures are followed and the recommended dosage of a particular drug is administered, then both nurses sign for the drugs in the patient's records. Drug usage is monitored by the charge nurse as the unit contains only small stocks of drugs, particularly controlled drugs such as

morphine. The charge nurse is responsible for ensuring that the use of such drugs is recorded, and for re-ordering such materials from the pharmacy.

If the nurse is unhappy with the diagnosis then she can refuse to administer the drug; within the health service it is impossible for anyone to be forcibly made to administer a drug. Should the nurse decide upon this course of action then she is required to report it to the sister in charge and the resolution of this conflict then becomes the responsibility of the sister in charge. Should the sister in charge be unable to resolve this situation then a higher authority may be invoked (the hospital's central administration) to deal with the situation, and it then becomes their responsibility. Within the accident and emergency department there are therefore two management structures; one for the doctors with the consultant at the head, and one for the nurses with the chief nurse at the head. These two management structures only come together in the central administration of the hospital.

8.3 An Information Model of the Current System

8.3.1 Introduction

In this section an information model of the current system will be developed and derived using the ideas and notations presented in this thesis. This model is then compared and contrasted with a model of the current system that was developed using standard business analysis techniques (IT-Uptake Project, 1986a; IT-Uptake Project, 1986b; Lorenzi et al., 1995) to see if the information model could: a) capture all the data contained in the standard business analyses, and b) show up any contradictions and anomalies.

8.3.2 An Enterprise Structural Model of the Current System

The purpose of the enterprise projection model is to identify and delineate the roles and functions performed by the various entities that exist within the A&E department. In defining these things the enterprise projection model is acting in such a way as to define the context within which the information model is to function. In Figure 8.3.2.1 the rectangles are used to represent agents and agency, while the rounded rectangles are used to depict the structural roles that a particular agent or agency may hold.

The structural relationships are illustrated by means of black lines linking together structural roles. This diagram is a very simple, cut down version of the complete enterprise diagram as it contains no functional roles or relationships. The

reason for this is that its purpose is simply to illustrate the roles and functions that are performed by the junior nurse. It acts in such a way as to set the context for any IT system that the junior nurse may need to use. It is from this form of diagram that a complete information projection diagram would be constructed. This diagram serves to define the contextual aspects of the information projection diagrams that will be presented later by defining the entities which exist within the system and the nature of the relationships that exist between the various entities.

In Figure 8.3.2.1 we can see the organisational structure for the A&E department depicted in Figure 8.2.3.1. In addition, we can also see what relationships the A&E department has with other departments within the hospital. In Figure 8.3.2.1 the junior nurse has two structural relationships with the chief nurse; the first is a server–client relationship representing the teacher–student relationship. The second is that of supervisor–subordinate which represents the managerial aspect of the chief nurse supervising the day to day functions performed by the junior nurse. The type of relationship between the junior nurse and the clerk is that of peer–coworker, where this relationship represents negotiation to access information on patients.

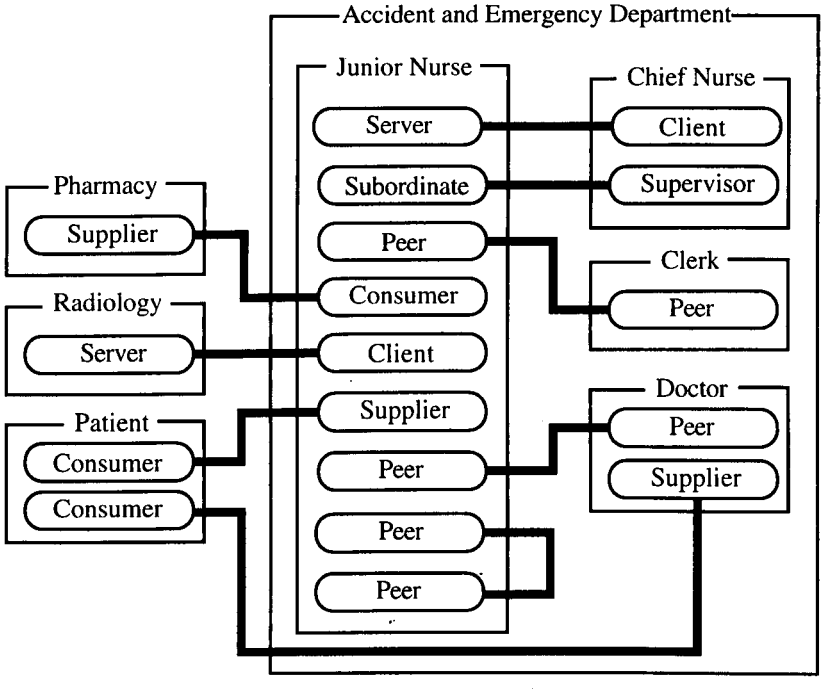


Figure 8.3.2.1 An Overview of the A&E Dept.

The peer relationship between the junior nurses depicted in Figure 8.3.2.1 tells us that one junior nurse has a peer relationship with every other junior nurse. The nature of the relationship between the pharmacy and the junior nurse is that of consumer–supplier, in that the nurse consumes and comments on items supplied by the pharmacy. While the nature of the relationship between the radiology department

and the nurse is that of client–server, the nurse can only request that the radiology department provide the x–rays for a particular patient. The nurse can not, however, request that a particular patient be x–rayed. The initiation on the conversation which would result in the radiology department x-raying a patient can only be done by a doctor or chief nurse.

The relationship between the doctor and the nurse is one of peer–peer. In this relationship the doctor engages the nurse in a conversation, the purpose of which is to arrange for the delivery of a medical service to a third party, that third party being the patient. However this relationship only extends to the medical duties performed by the nurse.

When the doctor requests that the nurse administer a drug the nurse must first be sure that the prescribed treatment for the patient is correct. The relationship between the patient and the nurse, and the patient and the doctor, is that of consumer–supplier as both the nurse and the doctor are deliverers of primary and secondary health care and the patient is the consumer.

8.3.3 Contractual Structural Models of the Current System

Figure 8.3.3.1 is derived from the Enterprise Projection Diagram that was presented in Figure 8.3.2.1. I have narrowed the focus of the diagrams down so that they now depict just three contractual relationships. The mapping between enterprise and information Projection Diagrams occurs at both syntactic and semantic levels (See Section 3.5 of Chapter 3). At the syntactic level we use the enterprise and information conceptual frameworks presented in Section 3.4 of Chapter 3. The first thing to note about the syntactic in this case study mapping is that the concepts of agents and agency have been mapped into the concept of parties. In addition, the concepts of structural and functional roles and relationships have been mapped into the concepts of contractual and conversational roles and relationships respectively. We a) map the agencies called Junior Nurse, Doctor and Patient into parties called Junior Nurse, Doctor and Patient, and b) map the structural relationship between the Junior Nurse and the Doctor into a contractual relationship.

At the semantic level we examine the nature of the relationships and make sure that the nature of the relationships is consistent between the various projection models. In addition, we examine associated invocation rights associated with the structural relationship. The contractual relationship is derived by comparing the nature of the structural relationship and its associated invocation rights with the various contractual relationships defined in Section 5.8.4 of Chapter 5. For example, in the

enterprise projection diagram the relationship between the doctor and the nurse is that of peer–peer. This relationship tells us that the doctor has no power over the junior nurse and consequently can not force the junior nurse to do anything or punish the junior nurse. As a result of this inability the doctor has to negotiate with the nurse for the delivery of treatment to the patient. This negotiation process can only be begun by the doctor and is started when the doctor makes a diagnosis on the patient. The doctor and nurse have to negotiate in order for the nurse to perform some service for the doctor. Due to the nature of the relationship between the doctor and the nurse, the contractual relationship that exists between them in the information projection is that of supplier–customer. The requester–provider conversational relationship that is associated with the doctor–nurse contractual relationship denotes the negotiation process by which the doctor attempts to persuade the nurse to perform a service. The requester–provider conversational relationship is derived directly in a one to one mapping from the functional relationship between the doctor and the nurse. The service that the nurse performs is the delivery of health care to a third agent, i.e. the patient. As a result of the delivery of this service both the doctor and the nurse discharge their obligations and in Figure 8.3.3.1 we can see that a Supplier–Consumer contractual relationship exists between the Doctor and the Patient. This relationship is derived from the structural relationship of Consumer–Supplier and the fact that either agency can invoke the delivery of the service of primary health care.

Figure 8.3.3.1 depicts the contractual and conversational relationships which will be examined in this case study. In this diagram the rectangles represent the parties that contain the contractual and functional roles. The contractual roles are connected together with lines of double thickness, while the conversational relationships are connected together with lines of single thickness. The contractual relationships act as a shorthand for the framework that permits and gives meaning to the conversational relationships, while the conversational relationships are used to delineate the types of conversations that two or more parties may engage in.

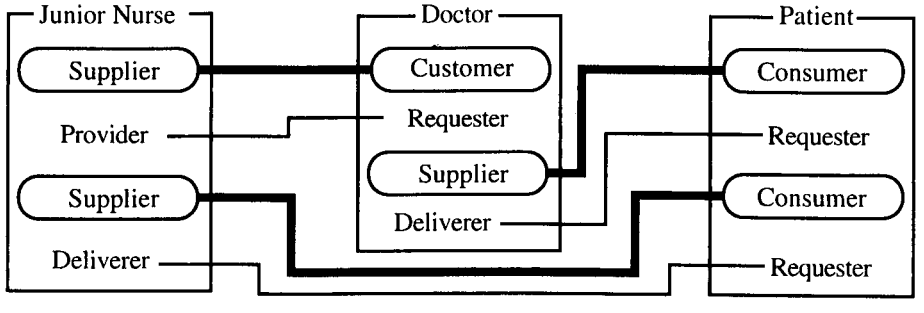


Figure 8.3.3.1 A Overview of the Contractual System

The first contractual relationship that will be examined and elucidated is that of the supplier–customer relationship that exists between the nurse and the doctor. The second contractual relationship that will be examined and elucidated is that of the supplier–consumer relationship that exists between the nurse and the patient.

In Section 5.6 of Chapter 5 the information modelling language is defined. In this language all the various objects and their relationships to other objects are defined. The rectangles depicted in Figure 8.3.3.2 denote the contractual roles held by the various parties, while the oval represents the conversation that the various parties may participate in. The rhomboids represent the information objects that are required by the conversation and contractual relationships to be present. The arcs represent the various relationships that can exist between the various objects. For example, the type of relationship between the supplier object and the diagnosis object in Figure 8.3.3.2 is that of read. On the arcs we can also place a conditional. The purpose of this conditional is to allow us to express the conditions under which a relationship is said to exist. If no conditional is expressed on an arc then the conditional is assumed to be true all the time and under all conditions.

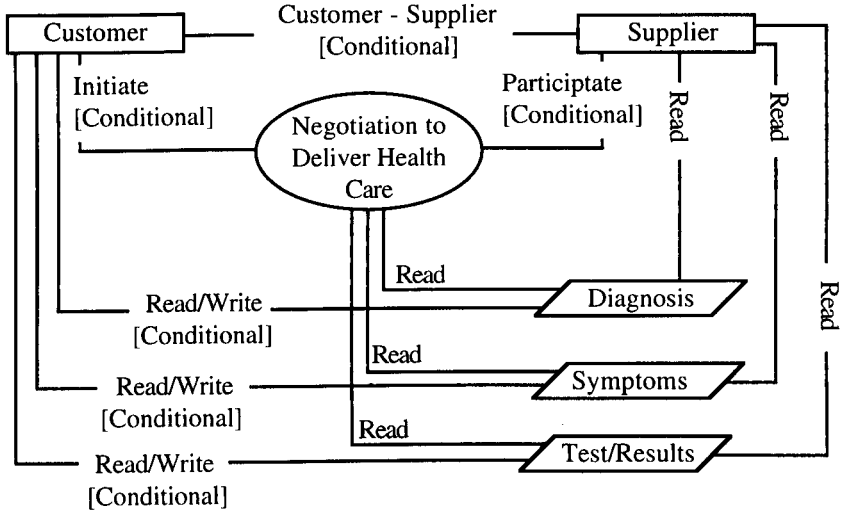


Figure 8.3.3.2 An Explication of the Doctor Nurse Contractual Relationship

In Figure 8.3.3.2 the doctor holds the customer contractual role while the junior nurse holds the supplier contractual role and the contractual relationship between them is one of customer–supplier. The doctor may initiate the conversation with the nurse provided that the doctor is treating the patient. The nature of this conversation is one of negotiation, as the doctor is attempting to convince the nurse to treat the patient in a prescribed manner. What makes this conversation interesting is that under Irish law the nurse is legally responsible for any treatment that she administers. As a result of this the nurse has to decide if she believes that the

treatment prescribed for the patient by the doctor is appropriate and adequate. In order for her to make a decision she has to perform a brief diagnosis based on the information that she has access to, and as a result of this a set of information access rights and requirements are generated.

Figure 8.3.3.2 depicts and delineates three very distinct information objects. The first object is used to hold the diagnosis of what ails the patient, while the second object is a record of the symptoms that are affecting the patient. The third and final information object is the result and order forms for any tests that have been carried out or ordered for the patient. By examining these objects and asking questions such as when, where and under what conditions are these objects created, modified and then destroyed, further requirements may be elicited. In Figure 8.3.3.2 the nurse is depicted as having only read access to the diagnosis, symptoms and test/results information objects. This level of access stems from the fact that the nurse has to form an opinion of what is wrong with the patient and consequently what the required cure is. In addition while she may examine the results of any tests that have currently been performed she does not have the authority to request that a new test is performed. The requesting of tests is a function that can only be performed by the doctors; this fact stems from the nature of the funding that the A&E department receives. The above diagram also represents the fact the doctor alone has read and write access to the various information objects from the perspective of this contractual relationship. None of the parties however have amend or alter access rights to the information objects so once a statement has been recorded it may never be unrecorded.

8.3.4 Analysis of Results

In this section I shall compare and contrast the information model of the current system with a model of the current system that was developed using various business analysis techniques (IT-Uptake Project, 1986a; IT-Uptake Project, 1986b). The purpose of this analysis is to ascertain whether or not the information model can: a) capture all the data contained in the standard business analyses, and b) show up any contradictions and anomalies.

In the business analysis presented in (IT-Uptake Project, 1986a; IT-Uptake Project, 1986b; Lorenzi et al., 1995) and in Section 8.2 the organisational structure is depicted by an organisational structure chart (Tosi,Rizzo, & Carroll, 1994). Within most business analysis techniques organisational structure charts are used to identify and validate potential stakeholders. Organisational structure charts allow us only to depict supervisor–subordinate relationships (See Figure 8.2.3.1). These types of charts only allow you to explore and analyse the organisation from an *is–supervised–by*

perspective. The organisational structure chart depicted in Figure 8.2.3.1 tells us that the student nurses and the staff nurses are supervised by the sister in charge, and that the senior house officers (SHOs) are supervised by the consultant. Organisational structure charts do not allow us to explore and define the nature of the relationships between the patient and the doctor, or the doctor and the junior nurse. Consequently they do not allow us to capture the nature of the information flows between the patient, the doctor and the nurse. As a result we have found diagrams like Figure 8.3.3.1 to be of more use than structure charts when conducting stakeholder analysis.

As we can see in Figure 8.3.3.2 the information models of the system allow us to capture, represent and validate back to the problem owners the contractual relationships that exist within the system. For example, we can observe in the information projection model shown in Figure 8.3.3.1 that the junior nurse has a supplier–customer relationship with the doctor, while in the enterprise projection model shown in Figure 8.2.3.1 the junior nurse has a peer–peer relationship with the doctor.

The reason for the difference between the enterprise projection model (Figure 8.2.3.1), and the information projection model (Figure 8.3.3.1), is that in the enterprise projection we are examining issues concerned with the mediation and fulfillment of responsibilities, while in the information projection we are examining issues concerned with the mediation and fulfillment of a contract. The enterprise projection model (Figure 8.2.3.1) tells us that the doctor has no authority over the junior–nurse, and that the doctor and the junior–nurse have to negotiate for the delivery of a service by the junior–nurse in order for the doctor to discharge a responsibility. The information projection model (Figure 8.3.3.1) tells us that the doctor and the junior–nurse have to negotiate for the delivery of a service by the junior–nurse. The contractual relationship between the doctor and the junior–nurse shows us that it is the doctor who initiates the conversation with the junior–nurse in order to get the junior–nurse to deliver a service.

From the analysis of this contractual relationship between the doctor and the junior–nurse shown in Figure 8.3.3.1 we can define a set of services that the junior nurse is required to support and the conditions and processes under which these services are delivered. In addition, such contractual models allow us to identify the access rights to the information that a doctor or nurse requires in order for them to deliver their service. For example, in Figure 8.3.3.2 we can see that the model captures the access rights that a nurse requires in order for the nurse to administer the prescribed drugs to a patient and so deliver primary health care. In allowing us to examine and classify the access rights over information, and the power relationships

between various parties, the information models have captured and represented all of the data that was captured by the various business analysis techniques and presented in (IT-Uptake Project, 1986a; IT-Uptake Project, 1986b; Lorenzi et al., 1995).

When examining Figure 8.3.3.2 we are prompted to ask about the conditionals associated with the relationships between objects. For example, Figure 8.3.3.2 prompts us to ask "Under what conditions can the customer and supplier participate in the negotiation for the delivery of primary healthcare?" The answer to this question does not lie in the data that has already been collected.

8.4 A Dynamic View of the Current System

In this section a dynamic model of the current system will be developed and examined to see if the dynamic model can: a) capture all the data contained in the various business analyses, and b) show up any contradictions and anomalies.

8.4.1 A Conversational View of the Current System

In this section the conversations that a nurse would engage in in order to administer some drugs to a patient at the request of a doctor will be modelled. In all of the following diagrams a rectangle is used to denote a speech act, and an isosceles triangle is used to signify a mental act of some description. In addition an oval is used to denote an instrumental act, and a rounded rectangle is used to express the starting or terminating points of the conversational unit. The notation is described and explained in detail in Section 6.2.

In Figure 8.4.1.1 the conversation begins with the doctor requesting the nurse to administer a drug to the patient. In so doing the doctor needs to have read and write access to all the patient's records so that they can be augmented by adding what ever drug is thought to be required. In requesting the nurse to administer a drug to the patient the doctor is asserting an informational attribute and mirroring this assertion in the patient's records. This informational attribute is the proposition that a particular drug is what the patient requires as part of his cure at this point in time.

The nurse now has to make a decision as to whether or not she is going to administer the drug. As under Irish Law an individual is legally responsible for any cure that they administer to a patient; this law consequently generates several direct and indirect requirements on the organisational information technology system.

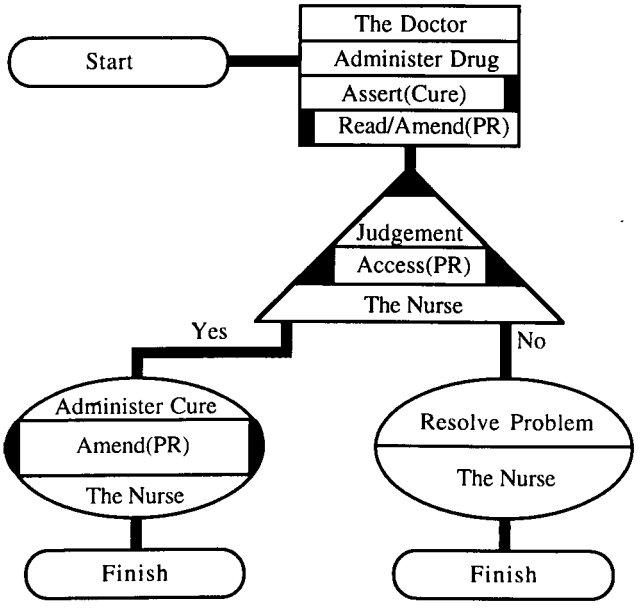


Figure 8.4.1.1 The Nurse's Decision

From the nurse's perspective, if the correct decision is to be made then access to all the informational objects and attributes related to the patient is required. The reason for this is so that the nurse may review them and form an opinion of what ails the patient so that a cure may be formulated and acted upon. In the above diagram the term *Access(PR)* is used to express the accessing of all the informational objects and informational attributes related to the patient. Having identified that the accessing of records is required, we may ask the question, "what is the nature of this accessing?" The answer is that the nurse may only read and that under these conditions write access is not allowed. Thus a nurse may read the result of a test that has been performed on the patient, but the nurse may not request that a particular test be performed.

We may also look at the mental act and ask the question, "what type of decision is it?" For example is it a decision for which a set of rules governing the outcome may be given, or is it a decision that calls upon the individual to make some kind of value judgement ? The latter kind is a mental act for which a computer may be used to aid in the making of, but it is undesirable to have a computer making by itself. The decision that the nurse is required to make is a value judgement due to legal aspects of the resulting action. For this reason the nurses have special educational needs. Consequently they are not only to be taught to apply a dressing or administer a drug; they are required to make informal diagnoses upon which they act. Thus they need to be taught how to make them and what information they need to make them.

The decision has two possible outcomes; the nurse can either choose to administer the drug, or the nurse can refuse. Because of the legal aspects of the system

the nurse can under no conditions be forced to administer the drug - should the nurse decide to refuse the doctor's request then steps must be invoked to resolve the conflict. We may point to the oval which represents the conversation that the nurse would engage in in order to resolve the conflict and ask the question, "What informational objects and attributes are required by this conversation?" From the perspective of the hospital these steps are formally defined and clearly stated for all to see. However, for the purpose of this thesis we will assume that the nurse decides to do as the doctor has requested.

In Figure 8.4.1.2 we can observe the legal implications of the administering of drugs and the way in which the hospital copes with them. It is hospital policy and operational procedure that one extra nurse is present when drugs are administered. The function of the second nurse is to check that the prescribed drugs are administered and that the correct procedures are followed. Both nurses are required to sign the drugs out in the drug trolley log book. In order for the second nurse to validate that the correct drugs are being signed out, the second nurse requires read access to the patients records. In addition, they are also required to record what drugs were administered, at what time and by whom and who was present.

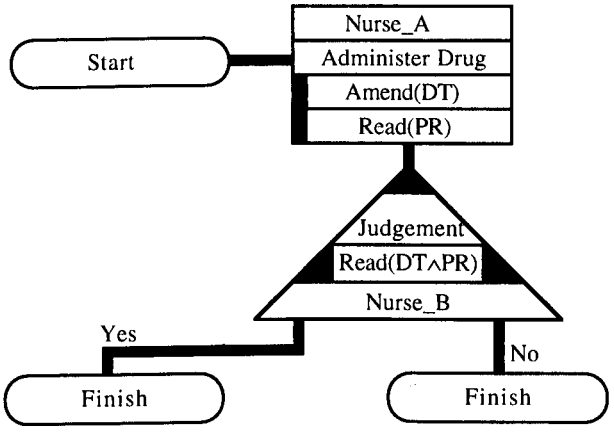


Figure 8.4.1.2 Administering the Correct Drug

In Figure 8.4.1.2 *Nurse_A* states what drug is planned to be administered, in what quantity and to which patient. The speech acts made by *Nurse_A* serve to act as an assertion of the nurse's intention. This assertion stems from the patient's records (PR), as a result the nurse is required to have read access to those records. *Nurse_B* is then required to make a decision, the nature of which is personal judgement. The nurse has to decide if *Nurse_A* has withdrawn the correct drug from the drug trolley (with reference to the patient's records and the nurse's assertion) and is going to administer the correct and prescribed amount. Consequently *Nurse_B* requires only read access to all informational objects and attributes associated with the patient. The

decision has only two possible outcomes, either the *Nurse_B* agrees with *Nurse_A* about the type and quantity of the drugs to be administered to the patient or *Nurse_B* does not agree. It is important to note that the nurse cannot under any circumstances postpone making the decision to some point in the future.

The question may be asked of the decision that the *Nurse_B* is making, "Can fixed rules be given so that the result of the decision may be determined?" If the answer to this question is yes then it is feasible to examine if a computer can make the decision. At first glance it may seem that the answer to the question is indeed yes, however upon careful examination of the role played by *Nurse_B* in administering drugs to the patient it soon becomes evident that the answer to the question is in fact no. The role of *Nurse_B* is to check and validate that *Nurse_A* has not made any mistakes; this checking process not only includes checking that the correct drug and dosage are administered, but also that they are administered correctly. It is because of the latter role that the nurse's decision cannot, and should not, be computerised. However, it is right and proper to now ask the question, "How may a computer assist the nurses in the performance of their duties?" The answer to this question will have direct influence on the organisational information technology system, and may be partially derived from examining how, when and where informational objects and attributes are manipulated.

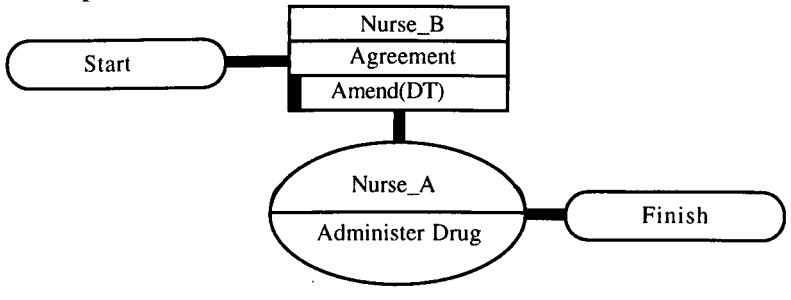


Figure 8.4.1.3 Nurse_B Agrees and Nurse_A Administers the Drug

Figure 8.4.1.3 depicts the course of events should *Nurse_B* agree with *Nurse_A* on the type and quantity of drugs to be administered to the patient. *Nurse_B* states agreement and amends the drug trolley records (DT) to indicate so. In addition this record also functions as a time stamp recording the time, place and the administerers of the treatment. This agreement acts as a signal that *Nurse_A* is free to administer the drug and consequently functions as a temporal constraint. *Nurse_A* then administers the drug to the patient under the watchful eye of *Nurse_B*. Once the administering of the drug has been executed then this conversation may terminate.

The administering of drugs to the patient is not a simple act as both nurses have to amend the patient's records to indicate the time, place and whether or not the drugs were administered. *Nurse_B* can always decide that *Nurse_A* is not

administering the drugs correctly and take steps to remedy the situation, these steps being formally laid down by the hospital. By examining the steps that *Nurse_B* would take to resolve the situation we can further elicit and capture requirements on the information technology system.

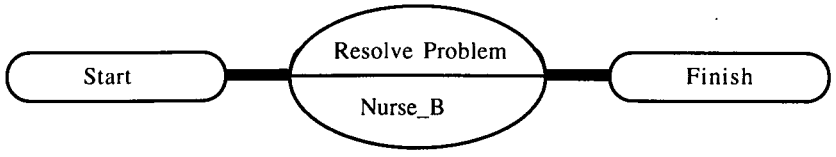


Figure 8.4.1.3 Conflict Resolution.

Figure 8.4.1.4 depicts the conflict resolution phase that *Nurse_B* may engage in on deciding that either the incorrect quantity or type of drug is going to be administered. This is the conversation that is executed if the result of the judgement depicted in Figure 8.4.1.2 is negative. In identifying that a conflict resolution phase is entered into we may ask the questions "what informational objects and attributes are needed to support this phase?" and "what access rights to informational objects and attributes are required ?" The answer to these questions will have a direct impact on the organisational information system that will be designed to aid the Accident and Emergency department in the performance of its duties.

8.4.2 The Hierarchy Coloured Petri Net Model

In this section of the case study a particular segment of the conversation that exists between the two nurses for the administration of a drug to a patient will be analysed. The segment that will be analysed using the coloured Petri net models is depicted in Figure 8.4.1.2. This analysis will take the form of that segment of the conversation being mapped down into a hierarchy net model and this simple net being analysed. Then the hierarchy net model will be mapped down into a low level coloured net model where more analysis will be performed.

The purpose of the hierarchical new model is to provide a high level formal description of the behaviour of the system to enable some simple analysis to be performed. The analysis that will be performed upon this net will attempt to answer questions such as "is the net reachable ?", "is the net bounded ?", "is the net live ?", "is the net persistent ?" For a full description of Petri net and coloured net analysis the reader is referred to (Murata, 1989) and (Jensen, 1992) respectively.

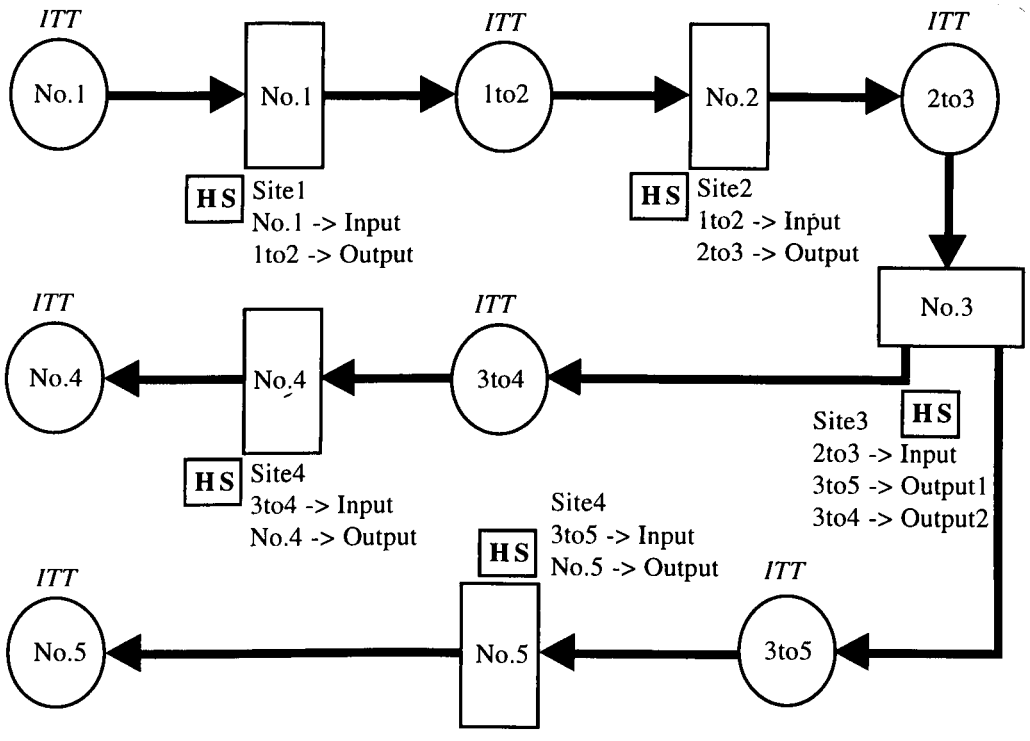


Figure 8.4.2.1 The Hierarchical Net Model

In Figure 8.4.2.1 the hierarchical coloured net model of the conversation described in Figure 8.4.1.2 is depicted. The circles represent places whilst the rectangles represent transitions. The first transition is called No.1 and this corresponds to the start act. The second transition is called No.2 and corresponds to the speech act while the third transition is called No.3 and corresponds to the decision act. The fourth and fifth transitions are called No.4 and No.5 respectively and correspond to the stop acts depicted in the conversation diagram.

For the purpose of this thesis and case study only two of the transitions depicted in Figure 8.4.2.1 will be mapped down into low level coloured nets and analysed further. However before this mapping and analysis is performed there is a certain amount of analysis that we can perform on the high level net description.

In Figure 8.4.2.1 the marking M_n is said to be reachable from a marking M_0 if there exists a sequence of firings that transform M_0 into M_n (Murata, 1989). Consequently the question is, for an inital marking M_0 in the net model depicted in figure 8.3.8.2.1, "is it possible to get to reach the other markings M_4 and M_5 through some series of transitions?" The marking M_0 is used to denote the state of a token in the place No.1, while the markings M_4 and M_5 are used to denote the states of tokens in the places No.4 and No.5 respectively. The simple answer to this question is yes, from the initial marking M_0 the markings M_4 and M_5 are reachable.

A petri net is said to be k-bounded or imply bounded if the number of tokens in each place does not exceed a finite number "k" for any marking reachable from M_0 (Murata, 1989). The net is also said to be safe if it is 1 bounded. The Petri net depicted in figure 8.4.2.1 is 1 bounded from an inital marking M_0 and it is therefore safe.

The question of liveness is one of the most important questions that we can ask of a net, as liveness is closely related to the complete absence of deadlocks in an operating system. A marking M_0 is said to be live if, no matter what other marking has been reached from M_0, it is still possible ultimately to fire any transition of the net by progression through some firing sequence. This means that a live Petri net guarantees deadlock–free operation, no matter what firing sequence is chosen (Murata, 1989). The answer to the question "is the Petri net depicted in Figure 8.4.2.1 live for marking M_0 live?", is yes, and thus the net is deadlock free.

A Petri net is said to be persistent if, for any two enabled transitions, the firing of one transition will not disable the other (Murata, 1989). Persistency is closely related to conflict free nets (Landweber & Robertson, 1978). For the net illustrated in Figure 8.4.2.1 we may assert that this net exhibits persistence.

In Figure 8.4.2.1, from the transition called No.3 it is possible to proceed down one of two paths. As a result the question has to be asked "what are the conditions under which the execution of a particular path is valid and what conditions result in the execution of that path?" The latter part of this question is concerned with knowledge and domain engineering as the question is concerned with eliciting a set of rules governing the outcome of the decision. Thus the question is concerned with elucidating the set of rules and guidelines that the nurse uses when making the decision. By examining how, when and where the nurse makes the decision we may arrive at the conclusion that the decision is probably deterministic. However, the rule set upon which the nurse operates when making the decision is unquantifiable.

8.4.3 The Low Level Coloured Petri Net Model

Having defined and analysed the hierarchical coloured petri net model of the conversation, it can now be mapped down into low level coloured nets and further analysed. The two low level nets that will be constructed for this case study are those for SITE2 and SITE3 (as defined in Figure 8.4.2.1).

The first site that will be analysed is that which corresponds to the speech act made by *Nurse_A*. This analysis will enable us to examine the nature of the state

change that occurs in the speech act. In addition, the pre, post, and invariant conditions will be formalised and examined for contradiction or omissions.

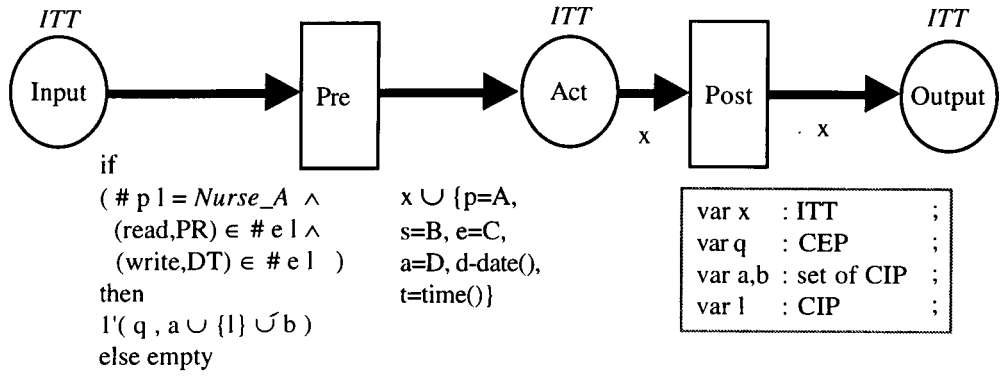


Figure 8.4.3.1 The Speech Act Petri Net Model

From the low level net model of the speech act depicted in Figure 8.4.3.1 we can observe that from the marking *input* the marking *output* is reachable. The reason for this is that there exists a sequence of firings such that a token can move from the *input* to the *output*. In addition, we may observe that while the net exhibits persistence, live, it is also 1 bounded and thus safe. This low level model is referred to as SITE2 in the hierarchical model.

What makes the low level net description of the speech act interesting is that it exhibits formal pre, post, and invariant conditions action statements. The question of what environmental conditions are required to exist in order for the speech act to be performed can be addressed. While the pre condition states that the nurse making the speech act is *Nurse_A*, it also states that read access to the patient's records and read write access to the drug trolley records is also required. The invariant and post conditionals are both simply true, in that no environmental or contextual conditions are required to hold in order for the action statements associated with the conditionals to be executed. The action associated with the invariant changes the state of the token by appending the speech act to the set of acts that comprise the token. This appending action is used to model and reason about the state changing aspects of the speech act.

As a result of this analysis we may assert that an information system that would hope to support the nurse in the execution of nursing duties would have to have access to both the patient and drug trolley records. In addition it would also have to support the concepts of traceability and authentication that currently exist in the paper based system.

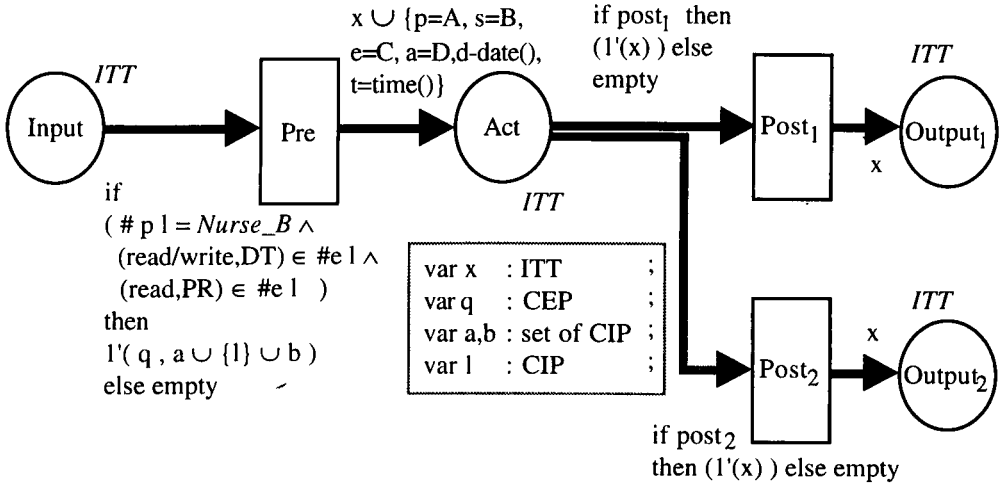


Figure 8.4.3.2 The Decision Act Petri Net Model

In Figure 8.4.3.2 the low level Petri net model of the decision act is depicted. This low level model is referred to as SITE3 in the hierarchical model. The analysis of this net leads us to the conclusion that the net is persistent, live, safe and that both of the output markings are reachable from the input marking. The persistence tells us that the Petri net, and therefore the conversation, is free of any conflicts. The L1-live tells us that the Petri net, and therefore the conversation, is free of deadlocks. The safety tells us that when the Petri net, and therefore the conversation, is being executed by a group, no other group may begin execution.

The precondition to the decision is that the decision belongs to *Nurse_B* and that the nurse must have read access to the patients records and read/write access to the drugs trolley records. Both of these requirements stem from the fact that the nature of the decision is one of verification. *Nurse_B* is verifying that *Nurse_A* has interpreted the patient's diagnosis correctly. The invariant conditional of the decision act is simply true under all conditions, and thus the action statements associated with it can simply be executed. The condition action invariant statements serve to provide a formal means with which to reason about the state changing aspects of the decision act.

The post condition action statements serve four very distinct functions. The first is to provide a formal framework from within which we may ask the question "when is this outcome of the decision valid?" So for example, we may ask the nurse if at all times and under all conditions both outputs from the decision are valid and the answer to this question is yes. The second function is to allow for the formalisation of the question "are these outputs from the decision the only output from the decision", and again the answer to this question from the nurse is yes. The third is to provide a formal framework from within which to examine if rules and procedures may be

given so that the decision may be deterministic. When we examine the nurse's answer to the question of whether or not the rules governing the outcome of the decision may be formalised and quantified, we arrive at the conclusion that the decision is probably deterministic. However the rules set upon which the nurse functions when performing the decision is unquantifiable. The fourth and final function is to provide a formal analysis tool so that we may examine if the disjunction of all the post conditions of the decision act holds. Should it not hold then we may ask the question "why not and is the reason for this known by, and accepted by, the problem owner?" It turns out to be a requirement on the nurse that the disjunction of both the post conditions holds at all points in time and under all conditions. This tells us that the nurse's decision can only ever have two outcomes. In addition, it is also a requirement that at any given point in time and space only one post condition may be valid. This means that the nurse can only ever choose one course of action. The nurse can either choose to agree with the diagnosed drug or not agree with the decision. The nurse can never however choose both.

8.4.4 Analysis of Results

The various analysis and data capture techniques that were applied to this case study present a reasonably detailed description of the process by which the doctor requests a certain treatment for a patient, and by which that treatment is delivered. In constructing and analysing Figure 8.4.1.1, Figure 8.4.1.2 and Figure 8.4.1.3 we can clearly see a) the process by which the doctor requests a certain treatment for a patient by performing a diagnosis, and b) that the treatment is delivered through a negotiation process with the nurse. In Figure 8.4.1.1, Figure 8.4.1.2 and Figure 8.4.1.3 we can see the representation of the data concerned with the delivery of treatment to a patient that was captured in the original analysis of the case study.

In constructing and analysing the explication of the contractual relationship between the doctor and the nurse (See Figure 8.3.3.2) we can define a) the information objects required by the nurse in order for the nurse to deliver the service, and b) the access rights of both the doctor and the nurse over the information objects. The information objects and their associated access rights were also captured in the original analysis of the problem. Figure 8.3.3.2 also prompts us to examine the conditionals under which the doctor can invoke the negotiation process and the conditionals under which the doctor and the nurse can access the information objects. These conditionals were not captured in the original analysis of the problem.

In Figure 8.4.1.1 we can see the doctor asserting what treatment is required by the patient, and the nurse making a decision about whether or not to administer the treatment. In constructing the conversation models we are forced to examine the

nature and types of the utterances made by the various parties involved. For example, in constructing the speech act made by the doctor we are forced to define what type of speech act it is, and what type of information objects are required in order for the doctor to make the utterance.

The conversation models also allow us to examine the decisions made during the execution of a conversation. For each decision we can ask "What are all the possible outcomes of this decision?", and "What information does the party making the decisions need in order to make the decision?" In examining the decision made by the nurse as to whether or not to administer the treatment to the patient we can clearly see that it is a personal decision and one which is not deterministic. We can also define information the nurse needs access to in order to make the decision. As a result of all of this direction when constructing the conversation models, the conversation models of the process by which the doctor requests a certain treatment for a patient, and that treatment is delivered, capture all of the data concerned with the process contained in the original analysis (See Section 8.2 of Chapter 8).

Conversation models also allow us to identify and validate stakeholders, as for each conversation model we need to identify the various parties involved and any other conversations that the conversation model may spawn. For example, in Figure 8.4.1.1 we can see that the nurse can refuse to administer a treatment to a patient. This prompts us to ask the question "How is the conflict resolved when the nurse does refuse?" The conflict is resolved through the use of an arbitrator, in this case the sister in charge. This arbitrator is obviously a stakeholder in the system and will require access to information contained in the information system. The normal information flow diagrams, like those depicted in Figure 7.2.2.2 and Figure 7.2.3.1 and used to analyse organisational behaviour, do not direct us towards identifying and validating additional stakeholders.

The construction and analysis of the coloured petri net model forces the formalisation of the conversation into predicate logic. This process of mapping from a conversation model of the system to a Petri net model of the system forces the problem owners to define formally conditions when certain actions may or may not be allowable. It is possible to derive a partial Petri net model from a conversation model (See Chapter 6). However, in order to complete the mapping process and produce an executable model the mapper is directed towards formalizing conditions governing the behaviour of the model. This formalisation process involves the problem solvers asking the problem owners more questions, and from these answers deriving and representing the formal conditions. The language used to express such conditions is predicate logic and predicate logic is something that most problem owners do not

understand. Consequently the problem owners can not verify that the formal model is in fact a correct representation of the real world situation. As a result if the Petri net models are used as part of a formal specification of the system it is possible for that formal specification of the system to be incorrect. It is felt that this inability to validate formally the petri net model of the system with the problem owners is a problem that needs to be addressed in the future.

The coloured petri net models of the conversation models were used to analyse and prove certain dynamic properties of the system. The coloured petri net models were used to examine properties such as safety, persistency, reachability and liveness. Its is believed that the formal analysis of such properties will allow us to show up possible contradictions and anomalies in their associated conversations.

The reachability property of a net tells us that every place in the net is reachable from the start state of the net. If our Petri net model of the conversations has this property then we may conclude that from the start of the conversation it is possible to reach every other state in the conversation. For example, if the conversation dealing with the resolution of conflict between the nurse and the doctor over the delivery of treatment to a patient were not reachable, then this would have serious implications for the nurse. It is therefore vital that we analyse conversations to identify such properties.

The property of liveness is closely related to the complete absence of deadlocks in an operating system. A net is said to be live if, no matter what marking has been reached, it is still possible to ultimately fire any transition of the net by progression through some firing sequence. If our Petri net model of the conversations has this property then we may conclude that from the start of the conversation it is not possible to reach deadlock. For example, if the conversation concerning the treatment of a patient was to deadlock then it is possible that the patient may die. It is for this reason that we want to prove that it is impossible to deadlock the system. From the analysis contained in section 8.4 we can prove that the conversations are both reachable and live, and this contains no contradiction or anomalies concerning reachability and liveness. The analysis of the petri net models contained in Section 8.4 of Chapter 8 also tells us that the nets are persistent and bounded.

8.5 Concluding Remarks on the Case Study

All of the models present in this chapter were presented back to the initial consultants at the Work Research Centre, Dublin, Ireland, who worked on the original problem in the ESPRIT IT–UPTAKE project. The purpose of the presentation was to

gather their comments, criticisms and suggestions concerning the modelling technique developed in this thesis. What follows is a summary of their comments.

As a consequence of using the organisation and information modelling techniques it became apparent that it would not be appropriate to computerize the accident and emergency (A&E) department of the hospital on its own. As the A&E department relies on so many other departments for services a computerized system could achieve its best results with these other departments being computerized.

The consultants found the enterprise structural diagrams (like those depicted in Section 7.3.1 of Chapter 7) and the contractual structural diagrams (like those depicted in Section 8.3.3 of Chapter 8) to be very expressive, easy to use, and easy to understand. They felt that the notation was simple enough, and expressive enough, and that problem owners would be able to understand it and use it themselves. They felt that the enterprise structural diagrams, and contractual structural diagrams provided an easy mechanism by which it is possible identify and validate potential stakeholders, and to define and analyse organisational boundaries. They felt that through the use of these diagrams it would be easy to explore with the problem owners, in real time, the possible implications of different organisational boundaries. The consultants felt that the concept of responsibility modelling was a major forward step and that it facilitated the identification of possible stakeholders and the unification of their views of the system. Out of all the types of models presented to the consultants they felt most positive about the enterprise structural diagrams and the contractual structural diagrams.

The consultants found the construction and analysis of explication of structural and contractual relationships (See Figure 8.3.2.2) to be useful in the identification of organisational resources. In addition, they felt that the identification of access modes and access rights and their associated conditions was a useful tool in the identification and validation of organisational policies. They also felt that the notation was simple enough, and expressive enough, and that the problem owners would be able to understand it and use it themselves.

The conversation models (See Section 8.4) received a mixed reception from the problem owners. They felt that the notation was expressive, simple to use and relatively simple to understand. They also liked the ability to identify the creation and manipulation of responsibilities and resources (See Figure 8.4.1.1). They felt that the notation allowed stakeholders to build up a consistent picture of a) what type of behaviour they were allowed to engage in, and b) when and where responsibilities were created and manipulated that bound to them. They felt that speech act theory was

very expressive and possibly a bit too complicated, but they accepted that it was a powerful concept and a very expressive tool. They felt that some of the points made by speech acts were very subtle but very important, and that some of the subtleties may be lost on people.

The consultants proved the most negative about the Petri net models. They acknowledged that proving certain properties like liveness and reachability were very desirable particularly in safety critical situations such as the Accident and Emergency department of a hospital. However they were very unsure of whether or not most problem owners would have the time or the inclination to construct and analyse such models. Most problem owners and problem solvers do not understand such formal languages, and it was therefore felt that the application of such models to all problems would be problematic. However they did acknowledge that there was a certain set of types of problems where the application of the Petri net modelling technique would be very useful and yield beneficial results. In addition, they felt that the hierarchical approach and use of coloured Petri nets was the best approach that they had seen to the problem of formalizing communication processes. They acknowledged that if the conversation governing the administration of drugs was to be computerised, then it is vital that the properties of liveness and reachability be formally proven.

The consultants felt very strongly that what the various modelling notations needed most was the support of a computer tool in constructing, validating and analysing the models. They felt that the tool should support the construction, simulation and analysis of the dynamic models of the system, and in particular the construction and analysis of the Petri net models of the system. The consultants stated that all the modelling notations required more heuristics (in particular the dynamic modelling notations) to aid in their construction and analysis. They acknowledged that the development of such heuristics would only happen in time as more people used the notations on a variety of different problems in a variety of different domains.

On the whole however the consultants welcomed the new modelling and analysis techniques and felt that they were a step forward.

Chapter 9 *Summary and Conclusions*

9.1 The Problem

Computers have evolved from being large cumbersome machines into small versatile machines in a very short space of time. With this rapid evolution their domain of application has grown until it now encompasses virtually every facet of daily life. Computers are being used now more than ever before within organisations as a means to maintain their competitive edge in a global marketplace. The more traditional methods have not fully realised the extent to which organisational information technology systems control and mediate social relationships (Dahlbom & Mathiassen, 1994), consequently a new approach is required.

The hypothesis presented in this thesis is that there is a set of organisational requirements on an information system which can usefully be identified and modelled by means of enterprise and information modelling architectures. The following assertions are made:

- That it is possible to create both an enterprise and an information modelling architecture, using social and linguistic constructs such that it is possible to map concepts between the two architectures. (1)
- That it is possible using these architectures to construct and validate the structural and dynamic enterprise and information models through a communicative dialectic process involving both the problem owners and the problem solvers. (2)
- The result of this mapping and modelling is that greater clarity and precision of the organisational context of the problem and its possible solutions can be developed. (3)

The validation of this thesis takes the form of two case studies. Both case studies are drawn from problems which real organisations are experiencing today. In addition, both case studies are drawn from different areas of industry so as to demonstrate domain diversity. The approach taken to ascertain whether greater clarity or precision can be achieved will be different for each case study, and each approach and case study is outlined below.

The first case study is taken from a regional electricity board (See Chapter 7). The electricity board is responsible for supplying electricity and electrical services to a large region, covering several large suburban and rural areas. The validation of the hypothesis against this case study will take the following two forms:

- An enterprise model of the current system was developed using the ideas and notations presented in this thesis. This model was then compared and contrasted with a model of the current system that was developed using various business analysis techniques (Eason, 1995; Eason, 1985; Eason, 1987).(See Section 7.3 of Chapter 7)

- An enterprise model of the system as proposed was developed and compared with the enterprise model of the current system. The purpose of this analysis was to ascertain whether the models contain any unexpected and significant results or differing views. The results of this analysis showed that the model of the system as proposed contained some unexpected and significant results (See Section 7.4 of Chapter 7) which serves to illustrate that the third bullet has been achieved.

The second case study is taken from an accident and emergency (A&E) department of an inner city hospital and was undertaken as part of the ORDIT project (See Chapter 8). This hospital is one of several that contains an A&E department in the wider metropolitan area and serves as a front line institution for the delivery of primary and secondary health care. The validation of the hypothesis against this case study employed the following three components:

- An information model of the current system is developed and derived using the ideas and notations presented in this thesis. This model is then compared and contrasted with a model of the current system which was developed using various business analysis techniques (IT-Uptake Project, 1986a; IT-Uptake Project, 1986b; Lorenzi,Riley,Ball, & Douglas, 1995). The analysis proved that the information model can: a) capture all the data contained in the various

business analyses, and b) show up any contradictions and anomalies. (See Section 8.3 of Chapter 8)

- A dynamic model of the current system is then developed and examined.The analysis proves that the dynamic model can: a) capture all the data contained in the various business analyses, and b) show up any contradictions and anomalies.(See Section 8.4 of Chapter 8)

- Both sets of models are then presented back to the analysts and consultants that originally worked on the problem and their comments and suggestions elicited and documented (See Section 8.5 of Chapter 8), and which serve to illustrate that the third bullet has been achieved.

9.2 Overview of the Thesis

Chapter one begins with a general problem definition statement of the problems concerned with requirements engineering. In this section the concept of problems as being social in disposition is introduced (Ehn, 1989; Ross, 1977). In the following section it is argued that the requirements engineering process is social in nature: when one is constructing a specification, one is constructing a social view of reality. Thus what is required of a requirements engineering method is that it be socially aware. The nature and purpose of a specification is explored in the next section. The concept of a specification as a communication medium (Zachman, 1987) is defined and explored. In the final sections I state what my hypothesis is and how I propose to prove it. A statement of the content of the thesis is given along with a summary of how and where the work contained in this thesis draws upon other work.

We start Chapter two with a set of classification and evaluation criteria with which the requirement engineering notations, techniques and methodologies will be graded. In the following section an overview of various methodologies such as SSM (Checkland, 1986), SADT (Ross & Schoman, 1977), SSADM (Nicholls, 1987) and CORE (Mullery, 1979) are presented. In addition, this section finishes with a general discussion on the methodologies currently available to an information technology analyst/engineer. The next section contains an overview of various requirements specification language and notations such as PSL/PSA (Teichroew & Hershey, 1977), HOS (Hamilton & Zeldin, 1976), RSL (Alford, 1977) and DFD (Dennis, 1974). This section terminates with a review of several other notation and languages not covered earlier in the chapter and a discussion of the various merits of the various languages and notations discussed.

Chapter three begins with an investigation into, and a classification of, the philosophical assumptions behind any modelling approach. In the following section the concepts of what an architecture is, and how it might be applied to the software engineering process is elucidated and developed (Perry & Wolf, 1992; Zachman, 1987). The structuring of an architecture to aid the process of requirements engineering is discussed and outlined in the next section. In the following section the concepts of enterprise and information modelling architecture are discussed and elucidated. In the next section a solution to the problem of mapping between these two modelling frameworks (architectures) is set forth. This solution is then used within a small example in the next section.

We start Chapter four with an examination of the organisational issues that current methods have failed to address adequately to some degree. The meaning and purpose of a method is then examined, along with the methodological, technical and sociological context within which the method defined in this thesis is required to function. The three main components that comprise the method are then defined with each one being expressed and discussed in turn, and the component relationships and interactions of the method are defined. In particular, attention is paid to the various components that make up the process model and the various projections that can be used within the process model to describe and analyse the system.

Chapter five begins with a discussion of what an organisational model is and how we might use it. The next sections describe in depth the various components and concepts that make up an enterprise projection diagram. The final sections describe in depth the numerous components and concepts that make up an information projection diagram.

Chapter six begins with a brief discussion and overview of the various notations that are currently available to model and analyse organisational dynamics. A graphical model of conversations which draws upon the concepts of speech acts and directed graphs is then presented. This is followed by an in depth overview of coloured Petri nets. A mapping from each component in the graphical conversation model to a coloured net construct is then defined along with a set of projection token types. Together both modelling techniques form a framework from within which organisational dynamics can be defined and reasoned about.

Chapter seven begins with a problem statement for the first case study (The Man in the Van). A set of enterprise diagrams is then constructed and analysed for the current system. A set of conversation diagrams is then derived from the perspective of a day electrician. The conversation diagrams are constructed so that the process of job

allocation and job completion can be analysed. From the conversation diagrams a picture of the flow and life cycle of the organisation's responsibilities is constructed and analysed. In addition, the objects and access rights over those objects needed by the electrician are identified. All of these models for the current system are then compared and contrasted with the models of the current system that were developed using various business analysis techniques. A set of enterprise and conversation models is then constructed for the system as proposed. Then the models of the current system are compared with the models of the proposed system in an attempt to ascertain whether the models contain any unexpected and significant results or differing views.

Chapter eight begins the second case study (The Jervis Street Hospital) with a problem statement. After this a small enterprise diagram of the organisation is constructed. From this enterprise diagram a contractual view of the system is derived and validated. The contractual model of the system is then compared and contrasted with a model of the current system that was developed using various business analysis techniques in an attempt to validate the modelling techniques. A set of conversation diagrams are then constructed for the junior nurse in the role of administerer of drugs. From these conversation diagrams a coloured Petri net model is derived and a set of formal requirements elucidated. These diagrams are then analysed to see if they contain any contradictions and anomalies. Finally the models created in the second case study are presented back to the original consultants who worked on the problem.

9.3 Concluding Remarks

By examining organisations and the sociological and cultural aspects of the way in which people work, the observation may be made that the organisation is an ever–changing and evolving entity. Organisations can be viewed as living entities that evolve, adapt and interact with their surroundings. Thus an information system becomes something that will either support or hinder this evolution. The question is "How does the work contained in this thesis help organisations to change and evolve for the better?" The answer is that the technique developed in this thesis addresses the issues of view diversity, view applicability, domain diversity, domain suitability, and traceability and there is guidance on its application.

View diversity is concerned with the creation, analysis, communication and unification of conflicting models of the system. It is through the application of view diversity that the problem owners and problem solvers communicate about the problem and its possible solution. The key to view diversity is that the problem owners and problem solvers use a shared common language. This language should allow for the expression and unification of conflicting statements. The English language supports

view diversity as through it we can create and analyse conflicting models of the system. We can also communicate our models between one another and thus resolve conflict and establish consensus between various view holders. View diversity is achieved within a modelling language only if that language allows for: a) multiple conflicting models of the system to created, analysed and unified, and b) the communication and resolution of conflicting models. The enterprise projection achieves view diversity through the application of the concept of responsibility as a tri-party relationship, and the modelling of the creation, manipulation and fulfillment of responsibilities. The information projection achieves view diversity through the use of contract modelling to define the context for propositions, and the modelling of the invocation and delivery of services.

In the Man in the Van case study (See Chapters 7) we can see differing enterprise projection models being generated and unified. In Chapter 7 two differing enterprise views are constructed and then compared and contrasted. The purpose of the notations is to identify areas of omission or conflict. The models then act as a medium of communication through the omission or conflict is communicated.The process of unification of these models is something that only the problem owners can do. The models identify and communicate omissions or conflicts – the problem owners and problem solvers then have to resolve these through discussion and dialogue. In Section 7.4 of Chapter 7, we can see that by bringing the models together we have identified several areas of conflict. As as result of this conflict the problem owners altered the proposed system so as to remove the conflicts.

View applicability is concerned with the creation, analysis, communication and unification of models that reflect a particular view of the system, where a particular view is applicable to a particular stakeholder in the system. It is through the application of view applicability that the problem owners and problem solvers communicate and establish a consensus of what the problem is and what a solution may look like. The key to view applicability is that the problem owners and problem solvers establish and validate a common language. Within the enterprise and information projection models view applicability is achieved through the use of social concepts within the modelling languages. The social concepts allow us to capture and validate the organisational context and value system that the information system is required to support. The enterprise projection achieves view applicability through the application of view diversity to the current problem owner. View applicability makes use of the concept of responsibility to define the view of the system from the perspective of the current problem owner. The information projection achieves view applicability by means of

modelling contracts and propositions from the perspective of the current problem owner.

Chapter 7 shows us the enterprise projection models for the system as it currently stands. In this model we can clearly see the use of responsibility as a relationship to capture the electrician's view of the problem. In the Hospital Accident and Emergency Department case study (See Chapter 8) we can see the modelling of contracts and propositions being used to capture the relationship between the doctor and the nurse. In both of these case studies we can see creation and analysis of models that reflect a particular view of the system, where a particular view is applicable to a particular stakeholder in the system.

Domain suitability is achieved through the use of linguistical, sociological and cultural components to a modelling language and an architecture. This objective is further supported by the realisation that the choice of an architecture is a cultural choice and therefore a socio–political statement. One only has to look at a typical medieval church to see how the architecture of the time embodies certain cultural aspects. The ability of an architecture to embody a set of cultural aspects is the key to its success. The architecture presented in this thesis has the ability to embody the organisational culture. This embodiment means: a) that the problem owners and problem solvers can see what values will be encapsulated in the system and how those values will be mediated, and b) that the problem solvers and problem owners can validate the values and their mediation and thus establish a shared understanding of what the problem is and what a solution may look like. This embodiment can be either descriptive or prescriptive in nature and as a result any organisation may embody its own social and cultural values and thus this approach achieves domain diversity and domain suitability through its generality.

Domain diversity is demonstrated by applying the technique to two different case studies. Each of the case studies is drawn from a different domain. The Man in the Van case study (See Chapter 7) is drawn from a service sector. The Hospital Accident and Emergency Department case study (See Chapter 8) is drawn from the Healthcare sector. Each of the two domains is different enough so that with the modelling technique being shown to be valid for both domains, the modelling technique is shown to have domain diversity.

In the enterprise projection models domain suitability is achieved through the modelling of the social concept of responsibility. In the enterprise models constructed in Chapter 7, we can see responsibility being modelled as a relationship, and the life cycle of these responsibilities being captured. In the beginning of Chapter 7, we can see the

responsibility that exists between a house–holder and the electricity–board being captured. Further on in Chapter 7 we can see the life cycle of the responsibility; we can see how this responsibility is created, flows through the electricity–board and is finally fulfilled. It is the ability to capture, represent and validate in any domain the responsibilities and their life cycles that gives the enterprise projection models domain suitability.

The information projection models achieve domain suitability through the modelling of contracts and from these models the derivation and specification of services. In Chapter 8, we can see the contract that exists between the doctor and the nurse, and from this contract we can see the service that the nurse is required to deliver. In examining the service we can also define how, and under what conditions the service is specified and delivered. It is the ability of the information project models to capture, represent and then validate back to the problem owners these issues that provide the information project models with domain suitability.

Traceability support is provided through the construction and association of a set of models to represent a given view on the system for a given holder of that view. Models can be used to enshrine a particular set of values or beliefs about the functioning or purpose of the organisation; this model can then be attached to the particular viewpoint holder on the system. Once a set of models has been constructed they can be analysed in an attempt to detect conflicts and omissions, and thus build consensus. Should some conflicts or omissions be detected then the models may be taken back to their respective owners and errors or omissions resolved. Consequently through the use of the models a consistent view of the organisational information technology system can be developed. This development process is dialectic in nature and provides us with traceability.

In both case studies we can see requirements being enshrined within the models. For example, in the Hospital Accident and Emergency Department case study (See Chapter 8) we see the requirement that the junior nurse needs access to the patient record's being embedded within the conversation models. Each model has an owner, where an owner is a person whose view of the system the model represents, and from each diagram we can derive a set of requirements. As a result we can now trace requirements by making them belong to a model or a set of models where models have owners. In addition, we can identify what requirements change as a result of changing a model.

Elicitation guidance comes in the form of relationships that bind objects within the modelling language together with a conceptual framework and a process model. The

relationships enforce strict type dependencies, which in turn serve to suggest and highlight areas where conflict or omissions lie. The conceptual framework provides a set of organisational conceptualisations from within which social, political and cultural aspects of the system may be structured and explored. This exploration takes the form of a dialectic process within which models are constructed and analysed and their implications explored. The process model identifies a set of stages within the requirements engineering process. These stages each possess a set of substages, each of which attempts to address and answer a particular question. As a result of this structuring questions can be identified and examined directly. The flexibility of the process model allows the problem owners and problem solvers to construct and utilise their own path through it.

9.4 Future Items of Research

This thesis has concentrated on the development of a socio-technical modelling framework from within which organisational information systems requirements may be elicited, represented, validated and verified. The purpose of this framework is to allow for and facilitate the encapsulation of cultural values and beliefs through the use of an architecture. The architecture facilitates the representation of organisational policies, goals and objectives. It is not claimed that the approach or framework presented in this thesis is the best solution to the problem of engineering organisational requirements. However the basic principles that underpin the approach could be built upon to provide a more complete approach for requirements engineering.

One way to check the usefulness of the framework for organisational requirements engineering presented in this thesis is to perform further case–studies. Within this thesis two worked examples are presented. Each example is drawn from a real problem which existed in the real world. Only the breadth of the problem domains has been altered so as to provide a problem which would fit within this thesis. Each of the examples draws upon different aspects of the framework in an attempt to illustrate the breadth and applicability of the framework.

The framework could be further developed in one of five directions: firstly a more detailed and complex case study could be undertaken. The case study would entail the modelling of both the organisational and information systems so as to examine how the two notations worked and interacted when applied to the same problem. In addition, the case study would serve to identify the ease with which it is possible to map concerns and issues from one notation into the next. This case study would function to identify the strength and weakness of the modelling framework and technique. In

addition, it would also function to develop a set of heuristics concerned with the application of the modelling technique.

The second direction for development is the expansion and evolution of the modelling architectures. This could take one of two forms; the first is for the expansion of the architectures to include the computation projection. This would allow for organisational issues to be elicited and mapped all the way down into computational objects so that their implications could be explored and expressed. The second is for the modelling notations themselves to be further expanded and unified across the application domains.

The third is to build up a method and associated heuristics so that the information contained in the conversation models can be mapped down into an executable behavioural model of the system and validated. The fourth direction of further research is to explore the way in which requirements change and evolve over a period of time. This exploration could include the idea of sensitivity analysis and context modelling. These two approaches could lead to a framework from within which changing requirements could be better understood and taken into account when developing a system.

The fifth and final direction of development is to expand the original technique itself so as to increase its level of integration with modelling techniques such as SSM (Checkland, 1986), ER models (Chen, 1976) and SADT (Ross & Schoman, 1977). This increased level of integration would provide the modelling technique developed in this thesis with the ability to share information. Thus problem solvers would be able to capture and utilise information about the organisational information system that we represented in other forms. This would have the effect of minimizing the time spent in capturing and representing data, and would maximize the data that we already have available.

Reference List

Agresti, W. W. (1986). What are the New Paradigms. In W. W. Agresti (Ed.), IEEE New Paradigms for Software Development, (pp. 6 - 11). IEEE Computer Society.

Alexander, C. (1979). The Timeless Way of Building. Oxford University Press.

Alford, M. W. (1977). A Requirements Engineering Methodology for Real-Time processing Requirements. IEEE Transactions on Software Engineering, SE-3(1), 60 - 69.

Alford, M. W.,Ansart, J. P.,Hommel, G.,Lamport, L.,Liskov, B.,Mullery, G. P., & Schneider, F. B. (1985). Chapter Three - Acquisition Environment. In Lecture Notes in Computer Science - Distributed Systems, Methods and Tools for Specification. An Advance Course 190. (pp. 45 - 130). Springer-Verlag.

Anderson, T. A., & Lee, P. A. (1990). Fault Tolerence Principles and Practice (Second ed.). Springer-Verlag.

ANSA (1990). The ANSA Reference Manual. Architectural Projects Management.

Anton, A. I.,McCracken, W. M., & Potts, C. (1994). Goal Decomposition and Scenario Analysis in Business Process Re-Engineering. In 6th Conference on Advanced Information Systems Engineering, . Utrecht, The Netherlands:

Ashworth, C. M. (1991). Structured Systems Analysis and Design Method (SSADM). In The Software Life Cycle (pp. 168 - 188).

Auramaki, E.,Lehtinen, E., & Lyytinen, K. (1988). A Speech Act Based Office Modelling Approach. ACM Transactions on Office Information Systems, 6(2), 126 - 152.

Austin, J. L. (1962). How to do things with words. Clarendon Press.

Avison, G. E., & Fitzgerald, G. (1988). Information Systems Development - Methodologies, Techniques and Tools. Blackwell Scientific.

Ball, M. J.,Douglas, J. V.,O'Desky, R. I., & Albright, J. W. (Ed.). (1991). Healthcare Information Management Systems: A Practical Guide. New York: Springer-Verlag.

Banton, M. (1965). Role, An Introduction to the Study of Social Relations. London: Tavistock.

Barr, A., & Feigenbaum, E. A. (1986). The Handbook of Artifical Intelligence. Addison-Wesley.

Barwise, J., & Perry, J. (1983). Situations and Attitudes. MIT Press.

Beck, K., & Cunningham, W. (1989). A Laboratory for Teaching Object Oriented Thinking. In Object-Oriented Programming Systems Languages and Applications (OOPSLA), (pp. 1 - 6). ACM Press.

Beer, S. (1981). The Brain of the Firm : the Managerial Cybernetics of the Organisation. J.Wiley.

BellCore (1991). Information Modelling Concepts and Guidelines (Technical No. SR-OPT-001826). Bell Communications Research.

Benner, K. M.,Feather, M. S.,Johnston, W. L., & Zorman, L. A. (1993). Utilizing Scenarios in the Software Development Process. In N. Prakash,C. Rolland, & B. Pernici (Ed.), Information System Development Process (A-30), . Como, Italy: North-Holland.

Benus, B.,Hoog, R. d., & Metselaar, C. (1993). Applying the CommonKADS Organisational Model No. KADS-II/TI.1/UvA/RR/004/4.1).

Beslmuller, E. (1988). Office Modelling Based on Petri Nets. In ESPRIT 1988 Part 1, (pp. 977 - 987). North Holland.

Biddle, B. (1979). Role Theory: Expectations, Identities and Behaviours. Academic Press.

Birtwistle, G.,Dahl, O. J.,Myhrhaug, B., & Nygaard, K. (1973). SIMULA. Petrocelli Charter.

Blair, G.,Gallagher, J.,Hutchison, D., & Sheperd, D. (1991). Object - Oriented Languages, Systems and Applications. Pitman.

Blazer, R., & Goldman, N. (1979). Principles of Good Software Specification and their Implications for Specification Languages. In Specifying Reliable Software, (pp. 58 - 67).

Blyth, A. (1994). Modelling and Eliciting Organisational and Information System Requirements for Medical Information Systems. In 18th Annual Symposium on Computer Applications in Medical Care, . Washington D.C., USA: Hanley and Belfus Inc. Medical Publishers.

Blyth, A. (1995a). Chapter 9: Requirements Engineering within the Fusion Method. In Fusion in the Real World Prentice-Hall.

Blyth, A. (1995b). Responsibility Modelling and Its Application to the Management of Change. Focus on Change Management - Cases in Business Process Re-Engineering, 17.

Blyth, A., & Chudge, J. (1993). The Role of Interaction Analysis in Requirements Engineering. In N. Prakash,C. Rolland, & B. Pernici (Ed.), IFIP WG 8.1 Information System Development Process, A-30 . Como, Italy: IFIP North Holland.

Blyth, A., & Chudge, J. (1994). Modelling Organisational Behaviour Using Social and Linguistic Constructs. In 23rd Annual Northeast Decision Science Conference, . Portsmouth, New Hamshire, USA:

Blyth, A., & Chudge, J. (1995). Modelling and Eliciting Organisational and Information System Requirements. In International Medical Informatics Association 8th World Congress on Medical Informatics (MEDINFO), . Vancouver, British Columbia, Canada:

Blyth, A.,Chudge, J.,Dobson, J., & Strens, R. (1993). ORDIT: A New Methodology to Assist in the Process of Eliciting and Modelling Organisational Requirements. In S. Kaplan (Ed.), Organisational Computing Systems (COOCS), . Milpitas, California, USA: ACM Press.

Blyth, A. J. C. (1989) Role Relation Diagrams. M.Sc., The University of Newcastle Upon Tyne.

Blyth, A. J. C.,Chudge, J.,Dobson, J. E., & Strens, M. R. (1992). The ORDIT Approach to Requirements Identification. In Computer Software and Applications Conference (Compsac), . Chicago: IEEE Computer Society.

Blyth, A. J. C.,Chudge, J.,Dobson, J. E., & Strens, M. R. (1993). A Framework for Modelling Evolving Requirements. In Computer Software and Applications Conference (Compsac), (pp. 83 - 89). Phoenix: IEEE Computer Society.

Blyth, A. J. C.,Chudge, J.,Dobson, J. E., & Strens, M. R. (1994). The ORDIT Approach to Organisational Requirements. In M.Jirotka & J.Goguen (Eds.), Requirements Engineering: Social and Technical Issues Academic Press.

Bodily, S. E. (1985). Modern Decision Making: A guide to Modelling with Decision Support Systems. Singapore: McGraw-Hill, Inc.

Boehm, B. W. (1976). Software Engineering. IEEE Transcations on Computers, C-25(12), 1226 - 1241.

Boehm, B. W. (1981). Software Engineering Economics. Prentice Hall.

Boehm, B. W. (1986). A Spiral Model of Software Development and Enhancement. ACM Software Engineering Notes, 11(4), 22 - 42.

Booch, G. (1991). Object Oriented Design with Applications. Benjamin/Cummings Publishing Company.

Brodie, M. L., & Mylopoulos, J. (1986). Knowledge Bases and Databases: Semantic vs. Computational Theories of Information. In New Directions for Database Systems (pp. 186 - 218).

Brooks, F. P. (1987). No Silver Bullet: Essence and Accidents of Software Engineering". IEEE Computer, 20(4), 10 - 19.

Cameron, J. R. (1986). An Overview of JSD. IEEE Transactions on Software Engineering, SE-12(2).

CCTTA (1995). Information Systems Procurement: The Total Acquisition Process (TAP) System Guide. London: Her Majesty's Stationary Office (HMSO).

Centre, N. I. M. (1993). The Common Basic Specification - Generic Model Version 2.0 No. NHS Information Management Centre.

Champeaux, D. d., & Faure, P. (1992). A Comparative Study of Object-Oriented Analysis Methods. The Journal of Object-Oriented Programming (JOOP), 5(1), 21 - 33.

Checkland, P. (1986). Systems Thinking, Systems Practice. Wiley.

Checkland, P., & Scholes, J. (1990). Soft Systems Methodology in Action. Chichester: Wiley.

Chen, P. P. (1976). The Entity-Relationship Model - Towards a Unified View of Data. ACM Transactions on Database Systems, 1(1), 9 - 36.

Christel, M. G., & Kang, K. C. (1992). Issues in Requirements Elicitation (Technical No. CMU/SEI-92-TR-12). Software Engineering Institute (SEI).

Chudge, J., & Fulton, D. (1994). Trust and Co-operation in Systems Development: Applying Responsibility Modelling to the Problem of Changing Requirements. In Proceedings of the Requirements Elicitation for Software-Based Systems Workshop University of Keele.

Chung, L. (1993). Representing and Using Non-Functional Requirements: A Process-Oriented Approach (Technical No. DKBS-TR-93-1). Department of Computer Science, University of Toronto.

Ciborra, R. D. (1987). Management Information Systems: A Contractual View. In Information Analysis: Selected Readings Addison Wesley.

Clocksin, W. F., & Mellish, C. S. (1981). Programming in Prolog. Springer-Verlag.

Coad, P. (1992). Object Oriented Patterns. Communications of the ACM, 35(9), 152 - 159.

Coad, P., & Yourdon, E. (1991). Object Oriented Analysis. Prentice Hall.

Coleman, D.,Arnold, P.,Bodoff, S.,Dollin, C.,Gilchrist, H.,Hayes, F., & Jeremaes, P. (1994). Object-Oriented Development: The Fusion Method. Prentice-Hall.

Commission, A. (1995). For Your Information: A Study of Information Management and Systems in the Acute Hospital. London: Her Majesty's Stationary Office (HMSO).

Conklin, E. J., & Yakemovic, K. C. B. (1990). Report on a Development Project Use of an Issue-Based Information System. In Computer Supported Co-operative Working, . Los Angles, California, USA.: ACM Press.

Conrath, D.,DeAntonellis, V., & Simone, C. (1988). A Comprehensive Approach to Modelling Office Organisation and Support Technology. In IFIP 8.4 Conference on Office Information Systems, .

Cunningham, R.,Finkelstein, A.,Goldsack, S.,Maibaum, T., & Potts, C. (1985). Formal Requirements Specification - The Forest Project. In IEEE International Workshop on Systems Specification and Design (IWSSD), . IEEE Computer Society Press.

Curtis, B.,Kellner, M., & Over, J. (1992). Process Modelling. Communications of the ACM (CACM), 35(9).

Czejdo, B.,Elmasri, R.,Embley, D. W., & Rusinkiewicz, M. (1990). A Graphical Data Manipulation Language for an Extended Entity-Relationship Model. IEEE Computer.

Dahlbom, B., & Mathiassen, L. (1994). Computers in Context: The Philosophy and Practice of Systems Design. NCC Blackwell.

Daly, E. (1977). Management of Software Development. IEEE Transactions on Software Engineering, SE-3(3), 229 - 242.

Dardenne, A. (1993). On the Use of Scenarios in Requirements Acquisition (Technical No. CIS-TR-93-17). Department of Computer and Information Science, The University of Oregon.

Dardenne, A.,Lamsweerde, A. v., & Fickas, S. (1993). Goal Directed Requirements Determination. Science of Computer Programming, 20(1-2), pp 3 - 50.

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

Davis, A. M. (1990). Software Requirements Analysis and Specification. Prentice Hall International.

Davis, R. (1983). Negotiation as a Metaphor for Distributed Problem Solving. Artificial Intelligence, 20, 63 - 109.

Dawson, S. (1986). Analysing Organisations. Macmillan Education.

Defence, M. o. (1991). Defence Standard 00-55: The Procurement of Safety Critical Software inDefence Equipement. (Interim Report No. Defence Standard). Ministry of Defence.

DeMarco, T. (1979). Structured Analysis and System Specification. Prentice Hall.

Dennis, J. B. (1974). First Version of a Data Flow Procedural Language. In Lecture Notes on Computer Science (pp. 362 - 376). Springer-Verlag.

Devlin, K. (1991). Logic and Information. Cambridge University Press.

Dobson, J., & McDermid, J. (1988). Security Models and Enterprise Models. In C. E. Landwehr (Ed.), Database Security, II (pp. 1 - 40). Kingston, Ontario, Canada: IFIP North-Holland.

Dobson, J. E. (1988a). Representing the Real World (Technical No. 267). The University of Newcastle Upon Tyne.

Dobson, J. E. (1988b). Security and Databases: A Methodological Approach (Technical No. 264). The University of Newcastle Upon Tyne.

Dobson, J. E., & McDermid, J. A. (1990). An Investigation into Modelling and Categorisation of Non-Functional Requirements (For the Specification of Surface Navel Command Systems) (Technical No. YCS 141). The University of York.

Dobson, J. E., & McDermid, J. A. (1991). An Investigation into Modelling and Categorisation of Non-Functional Requirements Part II: Methodology and Models (Technical No. YCS 160). Department of Computer Science, The University of York.

Eason, K. (1988). Information Technology and Organisational Change. Taylor and Francis.

Eason, K. (1995). Early Evaluation of the Organisational Implications of CSCW Systems. In P. Thomas (Eds.), CSCW Requirements and Evaluation Springer-Verlag.

Eason, K. D. (1985). Defining Information Technology Systems for the Electricity Supply Industry. In B. Shackel (Eds.), Human-Computer Interaction - INTERACT '85 (pp. 277 - 283). Amsterdam: North-Holland.

Eason, K. D. (1987). A User Centred Approach to the Design of a Knowledge-Based System. In H. J. Bullinger & B. Shackel (Eds.), Human-Computer Interaction - INTERACT '85 Amsterdam: North-Holland.

Ehn, P. (1989). Work-Oriented Design of Computer Artifacts. Arbetslivscentrum.

Ellis, C. (1979). Information Control Nets : A Mathematical Model of Office Information Flow. In Simulation, Measurement and Modelling of Computer Systems, (pp. 225 - 240). ACM Press.

Ellis, C. A., & Wainer, J. (1994). Goal-Based Models of Collaboration. Collaborative Computing, 1(1), 61 - 86.

Evangelist, M. (1986). Foundational Problems in Software Process Development. ACM Software Engineering Notes, 11(4), 17 - 20.

Fagan, M. (1974). Design and Code Inspection and Process Control in the Development of Programs (Technical No. IBM-SDD-TR-21-572). I.B.M.

Fairley, R. (1994). Risk Management for Software Projects. IEEE Software.

Fickas, S., & Anderson, J. S. (1989). A Prospective Shift: Viewing Specification Design as a Planning Problem. In International Workshop on Software Specification and Design (IWSSD), (pp. 177 - 184). IEEE Computer Society Press.

Finkelstein, A. (1989a). A Cooperative Framework for Software Engineering. In Hawaii International Conference on System Sciences (HICSS), 2 (pp. 189 -199).

Finkelstein, A. (1989b). Not Waving but Drowning: Representation Schemes for Modelling Software Development. In International Conference on Software Engineering (ICSE), (pp. 402 - 404). ACM Press.

Finkelstein, A., & Fuks, H. (1989). Multi - Party Specification. In Workshop on Software Specification and Design, (pp. 185 - 195). IEEE Computer Society Press.

Fishburn, P. C. (1970). Utility Theory for Decision Making. John Wiley & Sons, Inc.

Flynn, D. J. (1992). Information Systems Requirements: Determination and Analysis. London: McGraw-Hill.

Fox, R. (1995). No Laughing Matter. Communications of the ACM, 38(5), 10.

Freeman, P. A.,Sampaio, J. C., & Leite, P. (1991). Requirements Validation Through Viewpoint Resolution. IEEE Transactions on Software Engineering, SE-17(12), 1253 - 1269.

Freudenberg, W. R. (1992). Nothing Recedes Like Success? Risk Analysis and the Organisational Amplification of Risk. In Risk - Issues in Health abd Safety

Frost, R. A. (1986). Introduction to Knowledge Based Systems. London: Collins.

Galliers, R. D. (1987). An Approach to Information Needs Analysis. In R.D.Galliers (Eds.), Information Analysis: Selected Readings Addison Wesley.

Galton, A. (1987). Temporal Logics and their Applications. Academic Press.

GAO, U. S. G. A. O. :. (1979). Contracting for Computer Software Development: Serious Problems Require Management Attention to Aviod Wasting Additional Millions No. FGMSD-80-4). U.S. Government Accounting Office.

Gause, D. C., & Weinberg, G. M. (1989). Exploring Requirements: Quality Before Design. New York: Dorset House Publishing.

Genrich, H. J. (1986). Predicate / Transition Nets. In W. Brauer,W. Resig, & G. Rozenberg (Eds.), Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986 Part I Lecture Notes in Computer Science (pp. 207 - 247). Springer-Verlag.

Giddings, R. V. (1984). Accommodating Uncertainty in Software Design. Communications of the ACM, 27(4), 428 - 434.

Gladden, G. R. (1982). Stop the Life - Cycle, I want to Get Off. ACM Software Engineering Notes, 7(2), 35 - 39.

Goguen, J. A., & Burstall, R. M. (1977). Putting Theories Together to Make Specifications. In International Joint Conference on Artificial Intellegence, (pp. 1045 - 1058).

Goguen, J. A., & Burstall, R. M. (1986). The Semantics of Clear - A Specification Language. In Lecture Notes in Computer Science (pp. 292 - 332). Springer - Verlag.

Goguen, J. A., & Winkler, T. (1988). Introducing OBJ3 (Technical No. SRI-CSL-88-9). Stanford Research Institute (SRI).

Goldberg, A., & Robson, D. (1983). Smalltalk - 80 The Language and its Implementation. Addison Wesley.

Greenspan, S. J. (1984). Requirements Modelling: A Knowledge Representation Approach to Software Requiremnts Definition (Technical No. CSRG-155). Computer Stystems Research Group University of Toronto.

Griethuysen, J. J., & Jardine, D. A. (1988). Introduction to Infomod. Philips Applications and Software Servives — Philips International.

Guttag, J. V.,Horning, J. J., & Modet, A. (1990). Report on the Larch Shared Language Version 2.3 (SRC — Research No. 58). Digital Equipment Corporation.

Haack, S. (1978). Philosophy of Logics. Cambridge University Press.

Habermas, J. (1979). Communication and the Evolution of Society. Heinemann.

Hamilton, M. (1974). Higher Order Software Techniques Applied to a Space Shuttle Prototype Program. In Lecture Notes in Computer Science (pp. 17 - 31). Springer-Verlag.

Hamilton, M., & Zeldin, S. (1976). Higher Order Software - A Methodology for Defining Software. IEEE Transactions on Software Engineering, SE-2(1), 9 - 32.

Hamilton, M., & Zeldin, S. (1983). The Functional Life Cycle Model and its Automation: USE.IT. Journal of Systems and Software Science, 3(3), 25 - 62.

Harker, S. D. P., & Eason, K. D. (1985). Task Analysis and the Definition of User Needs (Technical No. 330). HUSAT Research Centre.

Harker, S. D. P.,Olphert, C. W., & Eason, K. D. (1990). The Development of Tools to Assist in Organisational Requirements Definition for Information Technology Systems. In INTERACT, (pp. 295 - 300).

Henderson-Sellers, B., & Edwards, J. M. (1990). The Object Oriented Systems Life-Cycle. Communications of the ACM, 09.

Hewitt, C. E. (1977). Viewing Control Structures as Patterns of Passing Messages. Artificial Intelligence, 8, 323 - 364.

Hewitt, C. E.,Bishop, P., & Steiger, R. (1973). A Universal Modular ACTOR formalism for Artificial Intelligence. In 3rd IJCAI, (pp. 235 - 245).

Holloway, C. A. (1979). Decision Making Under Uncertainty: Models and Choices. New jersey: Prentice Hall.

Holt, A. W. (1988). Diplans : A New Language for the Study and Implementation of Coordination. ACM Transactions of Office Information Systems, 6(2), 109 - 125.

Holtzblatt, K., & Beyer, H. R. (1995). Requirements Gathering: The Human Factor. Communications of the Association of Computing Machinery (CACM), 38(5), 31 - 23.

Hood, C. C.,Jones, D. K. C.,Pidgeon, N. F.,Turner, B. A., & Gibson, R. (1992). Risk Managment. In Risk: Analysis, Perception and Management London: The Royal Society.

Hsia, P.,Davis, A., & Kung, D. (1993). Status Report: Requirements Engineering. IEEE Software, 10(6), 75 - 79.

Hsia, P.,Samuel, J.,Gao, J.,Kung, D.,Toyoshima, Y., & Chen, C. (1994). Formal Approach to Scenario Analysis. IEEE Software, 11(2), 33 - 41.

IBM (1992). Business Systems Development Method: Introducing BSDM. London: IBM Corp.

IEEE (1984). IEEE Guide to Software Requirements Specifications: IEEE/ANSI Standard 830-1984. New York: Institute of Electrical and Electronic Engineers.

Ince, D., & Andrews, D. (1991). The Software Life Cycle. Butterworths.

IT-Uptake Project (1986a). Esprit Project 1030 - Human and Economic Factors in IT-Uptake Processes: Deliverable 1 No. Work Research Centre Ltd., Dublin.

IT-Uptake Project (1986b). Esprit Project 1030 - Human and Economic Factors in IT-Uptake Processes: Deliverable 2 No. Work Research Centre Ltd., Dublin.

Jackson, M. (1983). System Development. Prentice Hall.

Jahnichen, S. (1986). Towards an Alternative Model for Software Development. ACM Software Engineering Notes, 11(4), 66 - 70.

Jensen, K. (1986). Coloured Petri Nets. In W. Brauer,W. Resig, & G. Rozenberg (Eds.), Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986 Part I Lecture Notes in Computer Science (pp. 297 - 329). Springer-Verlag.

Jensen, K. (1992). Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use Volume 1. Springer-Verlag.

Johnson-Laird, P. N., & Byrne, R. M. (1991). Deduction : Essays in Cognitive Psychology. Lawrence Erlbaum Associates.

Katz, B., & Keshi, I. (1991). Speech Act Based Message Conversation System. In International Workshop on Computer Supported Cooperative Work, (pp. 59 - 73).

Keane, M., & Johnson, P. (1987). Preliminary Analysis for Design. In People and Computers III (pp. 133 - 146). Cambridge University Press.

Kingston, J.,Uschold, M.,Woherem, E., & Zorgios, Y. (1994). Enterprise State of the Art Survey: Enterprise Modelling Methods (Technical No. Ent/DE/1/1.0). The University of Edinburgh.

Kîtchenham, B. (1991). Chapter9: Making Process Predicitions. In N. E. Fenton (Eds.), Software Metrics: A Rigorous Approach (pp. 132 - 149). Chapman & Hall.

Klein, H. K., & Hirschheim, R. A. (1987). A Comparative Framework of Data Modelling Paradigms and Approaches. The BSC Computer Journal, 30(1), 8 - 15.

Kolf, F., & Oppelland, H. J. (1983). Design Oriented Approach in Implementation Research. The Project PORGI. In Human Factors in Computing Systems, 26 . ACM Press.

Lamb, S. S.,Leck, V. G.,Peters, L. J., & Smith, G. L. (1978). SAMM: A Modelling Tool for Requirements and Design Specification. In Computer Software and Applications Conference (Compsac), (pp. 48 - 53). IEEE Computer Society Press.

Lambert, R., & Peppard, J. (1993). Information Technology and New Organisational Forms: Destination But No Road Map. The Journal of Strategic Information Systems: Incorporating International Information Systems, 2(3), 180 - 205.

Landweber, L. H., & Robertson, E. L. (1978). Properties of Conflict Free and Persistent Petri Nets. Journal of the ACM, 25(3), 352 - 364.

Lee, R. M. (1988). Bureaucracies as Deontic Systems. ACM Transactions on Office Information Systems, 6(2), 87 - 108.

Lee, S., & Sluizer, S. (1991). An Executable Language For Modelling Simple Behaviour. IEEE Transactions on Software Engineering, SE-17(6), 527 - 543.

Leiberman, H. (1987). Concurrent Object Oriented Programming in ACT 1. In O. O. C. Programming (Eds.), (pp. 9 - 36). MIT Press.

Londeix, B. (1987). Cost Estimation for Software Development. Addison-Wesley.

Longworth, G. (1989). Getting the System You Want - A User's Guide to SSADM. NCC Publications.

Lorenzi, N. M.,Riley, R. T.,Ball, M. J., & Douglas, J. V. (Ed.). (1995). Transforming Health Care Through Information Case Studies. Springer-Verlag.

Lucas, P. (1987). VDM: Origins, Hopes and Achievements. In Lecture Notes in Computer Science (pp. 1 - 18). Springer-Verlag.

Lunberg, M. (1982). The ISAC Approach to Specification. In Information Systems Design Methodologies: Improving on Practice North Holland Press.

Lyytinen, K. (1987a). Different Perspectives on Information Systems: Problems and Solutions. ACM Computing Surveys, 19(1), 5 - 46.

Lyytinen, K. (1987b). Two Views of Information Modelling. Information and Management, 12, 9 - 19.

Lyytinen, K.,Auramaki, E., & Hirschheim, R. (1991). Modelling Offices Through Discourse Analysis: The SAMPO Approach. The Computer Journal, 35(4), 342 - 352.

Lyytinen, K.,Auramaki, E., & Hirschheim, R. (1992). Modelling Offices Through Discourse Analysis: A Comparison and Evaluation of SAMPO with OSSAD and ICN. The Computer Journal, 35(5), 492 - 500.

MacLean, R.,Stepney, S.,Smith, S.,Gradwell, D.,Hoverd, T.,Katz, S., & Tordoff, N. (1994). Analysing Systems: Determining Requirements for Object-Oriented Development. Prentice-Hall.

Martin, M. (1991). Political Systems. The British Journal of Healthcare Compting, March, 31 - 33.

Martin, M., & Oswald, G. (1992). Modelling the Case Studies. In R. J. D. Power (Eds.), Cooperation Among Organisations: The Potential of Computer Supported Cooperative Work (pp. 41 - 71). Springer-Verlag.

Masini, G.,Napoli, A.,Colnet, D.,Leonard, D., & Tombre, K. (1989). Object Oriented Languages. Academic Press.

Mason, R. O., & Mitroff, I. T. (1981). Challenging Strategic Planning Assumptions. New York: John Wiley and Sons.

McCracken, D. D., & Jackson, M. A. (1982). Life Cycle Concepts Considered Harmful. ACM Software Engineering Notes, 7(2), 29 - 32.

McDermid, J. A., & Dobson, J. E. (1989). Security Models and Enterprise Models (Technical No. YCS111). University of York - Department of Computer Science.

McLean, J. (1984). A Formal Method for the Abstract Specification of Software. Journal of the Association for Computer Machinery, 13(3), 600 - 627.

McLean, J. (1990). Security Models and Information Flow. In IEEE Symposium on Security and Privacy, (pp. 180 - 187). IEEE Computer Society Press.

Mills, H. D.,Linger, R. C., & Hevner, A. R. (1986). Principles of Information Systems Analysis and Design. Academic Press.

Minsky, M. (1975). A Framework for Representing Knowledge. In The Psychology of Computer Vision (pp. 211 - 281). McGraw-Hill.

Moffett, J. D., & Sloman, M. S. (1992). Policy Hierarchies for Distributed Systems Management (Domino No. Arch/IC/6).

Moore, A. (Ed.). (1993). Meaning and Reference. Oxford, UK: Oxford University Press.

Morris, P.,Coombes, A., & McDermid, J. (1994). Requirements and Traceability. In K. Pohl,G. Starke, & P. Peters (Ed.), Requirements Engineering: Foundations of Software Quaility, . Utrecht, Netherlands: Aachener Beitrage Zur Informatik.

Mullery, G. P. (1979). CORE - A Method for Controlled Requirement Specification. In International Conference on Software Engineering (ICSE), (pp. 126 - 135). IEEE Computer Society Press.

Mumford, F., & Hedberg, B. (1975). The Design of Computer Systems. In IFIP Human Choice and Computers, . North-Holland.

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

Myers, G. J. (1975). Reliable Software Through Composite Design. New York: Wiley.

Mylopoulos, J.,Chung, L.,Yu, E., & Nixon, B. (1993). Requirement Engineering 1993: Selected Papers (Technical No. DKBS-TR-93-2). Department of Computer Science, University of Toronto.

National-Research-Council (1983). Risk assessment in the Federal Government: Managing the Process No. National Research Council, Washington D.C., National Academy Press.

Neumann, P. G. (1982). Psychosocial Impplications of Computer Development and Use: Zen and the Art of Computing. ACM Software Engineering Notes, 7(2), 3 - 12.

Newell, A., & Simon, H. A. (1972). Human Problem Solving. Englewood Cliff.

Nicholls, D. (1987). Introducing SSADM - The NCC Guide. NCC Publications.

Nichols, G. E. (1987). On the Nature of Management Information. In Information Analysis: Selected Readings (pp. 7 - 18). Addison Wesley.

Noble, F. (1991). Seven Ways to Develop Office Systems: A Managerial Comparison to Office System Development Methodologies. The Computer Journal, 34(2), 113 - 121.

Oberquelle, H. (1998). Role/Function/Action-Nets as a Visual Language for Cooperative Modelling. In IFIP International Workshop on Human Factors of Information Systems Analysis and Design, .

Ohno, Y., & Rzepka, W. (1985). Requirements Engineering Environments: Software Tools for Modelling User Needs. IEEE Computer, 18(4).

Ould, M., & Huckvale, T. (1994). Business Process Modelling: Process Modelling What, What and How No. Praxis Plc.

Parker, M. M.,Benson, R. J., & Trainor, H. E. (1988). Information Economics: Linking Business Performance To Information Technology. Prentice-Hall International.

Pava, C. (1983). Managing New Office Technology: An Organisational Strategy. New York: Free Press.

Perry, D. E., & Wolf, A. L. (1992). Foundations for the Study of Software Architecture. ACM Software Engineering Notes, 17(4), 40 - 52.

Petri, C. A. (1966). Kommunikation mit Automaten (Technical No. RADC-TR-65-377). Griffiths Air Force Base.

Peugeot, C., & Franckson, M. (1991). Specification of the Object and Process Modelling Language No. ESF Report No. D122-OPML-1.0).

Potts, C. (1989). A Generic model for representing Design Methods. In 11th International Conference on Software Engineering, . IEEE Computer Press.

Potts, C. (1991). Seven (Plus or Minus Two) Challenges for Requirements Research. In International Workshop on Software Specification and Design (IWSSD), (pp. 256 - 259). IEEE Computer Society Press.

Potts, C.,Finkelstien, A.,Aslett, M., & Booth, J. (1986). Structured Common Sense: A Requirements Elicitation and Formalization Method for Modal Action Logic No. FOREST Deliverable.

Power, R. J. D. (Ed.). (1993). Cooperation Among Organisations. Springer-Verlag.

Quintus (1990). Qunitus Flex 1.21 Reference Mannual. Quintus Corporation.

Randell, B.,Laprie, J. C.,Kopetz, H., & Littlewood, B. (Ed.). (1995). Predictable Dependable Computing Systems. Springer-Verlag.

Randell, B.,Lee, P. A., & Treleaven, P. C. (1978). Reliability Issues in Computing Systems Design. ACM Computing Surveys, 10(2), 123 - 165.

Rein, G. L., & Singh, B. (1992). Role Interaction Nets (RINs): A Process Description Formalism (Technical No. CT-083-92). Micro-Electronics and Computer Technology Corporation (MCC).

Reiter, R. (1985). A Logic for Default Reasoning. In Readings in NonMonotonic Reasoning (pp. 68 - 93). Morgan Kaufmann.

Rivett, P. (1980). Modelling Building for Decision Analysis. John Wiley and Sons Ltd.

Robinson, W. N. (1989). Integrating Multiple Specifications Using Domain Goals. In International Workshop on Software Specification and Design (IWSSD), (pp. 219 - 226).

Robinson, W. N. (1990). Negotiation Behaviour During Requirements Specification. In International Workshop on Software Engineering, (pp. 268 - 276). IEEE Computer Society Press.

Rolland, C. (1994). A Contextual Approach for the Requirements Engineering Process. In 6th Intl. Conference on Software Engineering and Knowledge Engineering, . Jurmala, Latvia:

Roman, G. C. (1985). A Taxonomy of Current Issues in Requirments Engineering. IEEE Computer, 18(4), 14 - 21.

Rose, T.,Jarke, M.,Gocek, M.,Maltzahn, C.; & Nissen, H. (1991). A Decision-Based Configuration Process Environment. Software Engineering Journal, 6(5).

Ross, D. T. (1977). Guest Editorial - Reflections on Requirement. IEEE Transactions on Software Engineering, SE-3(1), 2 - 5.

Ross, D. T., & Schoman, K. (1977). Structured Analysis for Requirements Definition. IEEE Transactions on Software Engineering, SE-3(1), 6 - 15.

Rumbaugh, J.,Blaha, M.,Premerlani, W.,Eddy, F., & Lorensen, W. (1991). Object Oriented Modelling and Design. Prentice Ahll.

Ryan, G. M.,Wynne, R. F.,Cullen, K. M.,Ronayne, T. R., & Dolphin, C. (1988). Human and Economic Factors in the Uptake Processes. Volume 1. Concepts and Methodology for Understanding and Managing IT Uptake Processes in User Organisations (Technical No. ESPRIT Project 1030). Work Research Centre.

Schank, R. C. (1972). Conceptual Dependency: A Theory of Natural Language Understanding. Cognitive Psychology, 3, 552 - 631.

Schank, R. C., & Abelson, R. P. (1977). Scripts, Plans and Knowledge. In Thinking: Readings in Cognitive Science

Searle, J. R. (1969). Speech Acts - An Essay in the Philosophy of Language. Cambridge University Press.

Searle, J. R. (1979). Expression and Meaning. Cambridge University Press.

Searle, J. R., & Vanderveken, D. (1985). Foundations of Illocutionary Logic. Cambridge University Press.

Senge, P. M. (1990). The Fifth Discipline: The Art & Practice of The Learing Organisation. New York:

Sievert, G. E., & Mizell, T. A. (1985). Specification - Based Software Engineering with TAGS. IEEE Computer, 18(4), 56 - 65.

Silverberg, B. A.,Robinson, L., & Levitt, K. N. (1979). The HDM Handbook (Project No. No. 4828). Stanford Research Institute.

Singh, B. (1992). Interconnected Roles (IR): A Coordination Model (Technical No. CT-084-92). Micro-Electronics and Computer Technology Cororation - MCC.

Sodily, S. E. (1985). Modern Decision Making: A Guide to Modelling with Decision Support. Singapore: McGraw-Hill.

Somerville, I. (1985). Software Engineering (Second ed.). Addison-Wesley.

South West Thames Regional Health Authority, S. (1993). Report of the Inquiry into the London Ambulance Service No. Communications Directorate.

Spivey, J. M. (1989). The Z Notation - A Reference Manual. Prentice Hall.

Stamper, R. (1985). Management Epistemology: Garbage In, Garbage Out. (And What About Deontology and Axiology ?). In Knowledge Representation for Decision Support Systems (pp. 55 - 77). Elsevier Science.

Stephens, S. S., & Tripp, L. L. (1978). Requirements Expression and Verification Aid. In International Conference on Software Engineering - ICSE, (pp. 101 - 108). IEEE Computer Society Press.

Strens, M. R., & Sugden, R. C. (1995). Criteria for the Assessment of Representation Methods as Vehicles for Handling Change. In IEEE International Symposium on Systems Engineering of Computer Based Systems (ECBS), . Tucson, Arizona: IEEE Computer Press.

Strens, R., & Dobson, J. (1994a). Organisational Definition for Information Technology. In International Conference on Requirements Engineering, (pp. 158 - 165). Colorado Springs, USA: IEEE Computer Society Press.

Strens, R., & Dobson, J. E. (1994b). Responsibility Modelling as a Technique for Requirements Definition. Intelligent Systems Engineering, 3(1), 20 - 26.

Sutcliffe, A. G., & Maiden, N. A. M. (1993). Bridging the Requirements Gap: Policies, Goals and Domains. In Seventh International Workshop on Software Specification and Design (IWSSD), . Redondo Beach, California: IEEE Computer Press.

Tardieu, H. (1992). Issues for Dynamic Modelling through Recent Developments in European Methods. In H. G. Sol & R. L. Crosslin (Eds.), Dynamic Modelling of Information Systems (pp. 3 - 24). North-Holland.

Tarski, A. (1956). Logic, Semantics, Metamathematics: Papers from 1923 to 1938. Claredon Press.

Teichroew, D., & Hershey, E. A. (1977). PSL/PSA: a computer approach to information processing systems modelling. IEEE Transactions of Software Engineering, SE-3(1), 41 - 48.

Thomas, E. J., & Biddle, B. J. (1979). The Nature and History of Role Theory. In Role Theory: Concepts and Research (pp. 3 - 19). Frieger.

Tosi, H. L.,Rizzo, J. R., & Carroll, S. J. (1994). Managing Organisational Behaviour. Blackwell Publishers.

Trattnig, W., & Kerner, H. (1980). EDDA, A Very-High-Level Programming and Specification Langauge in the Style of SADT. In Computer Software and Applications Conference (Compsac), (pp. 436-443). Chicago, USA: IEEE Computer Society Press.

Tse, T. H., & Pong, L. (1991). An Examination of Requirements Specification Languages. The Computer Journal, 34(2), 143 - 152.

Tsichritzis, D. C., & Lochovsky, F. H. (1982). Data Models. Prentice Hall.

Ullman, J. D. (1988). Data Models for Database Systems. In Database and Knowledge - Base Systems. Volume One (pp. 34 - 42).

Veryard, R. (1992). Information Modelling: Practical Guidance. Prentice Hall.

Wiener, L. R. (1993). Digital Woes: Why We Should Not Depend on Software. Addison-Wesley.

Wileden, J. C. (1986). This is IT: A Meta-Model of the Software Process. ACM Software Engineering Notes, 11(4), 11 -1 6.

Wilks, Y. A. (1977). Methodological Questions about Artificial Intelligence: Approaches to Understanding Natural Language. Journal of Pragmatics, 1, 96 - 84.

Wing, J. M.,Guttag, J. V., & Horning, J. J. (1985). The Larch Family of Specification Languages. IEEE Software, 2(5), 24 - 36.

Winograd, T., & Flores, F. (1987). Understanding Computers and Cognition. Addison-Wesley.

Wittgenstein, L. (1958). Philosophical Investigations. Oxford: Basil Blackwell.

Wittgenstein, L. (1961). Tractatus Logico Philosphicus. London: Routledge and Kegan Paul Ltd.

Woodcock, J. C. P., & Brien, S. M. (1991). W: A logic for Z. In The Annual Z User Meeting, . York:

Woolridge, M., & Jennings, N. (1995). Agent Theories, Architecture and Languages: A Survey, Intelligent Agents. In M. Woolridge & N. Jennings (Eds.), ECAI-94 Workshop on Agent Theories, Architectures, and Languages: Number 890 in Lecture Notes in Computer Science (pp. pp.1-39). Amsterdam, The Netherlands: Springer-Verlag.

Wunderlich, D. (1979). Foundations of Linguistics. Cambridge University Press.

Wynn, E. H. (1979) Office Conversations as an Information Medium. Ph.D., Department of Anthropology, University of California, Berkeley.

Yeh, R. T. (1982). Requirements Analysis - A Management Perspective. In Computer Software and Applications Conference. (Compsac), (pp. 410 - 416). IEEE Computer Society Press.

Yourdon, E. (1975). Techniques of Program Structure and Design. Prentice-Hall.

Yu, E. S. K., & Mylopoulos, J. (1994). Using Goals, Rules and Methods to Support Reasoning in Business Process Re-Engineering. In Twenty-Seventh Hawaii International Conference on System Sciences, .

Zachman, J. A. (1987). A Framework for Information Systems Architecture. IBM Systems Journal, 26(3), 276 - 292.

Zachman, J. A., & Sowa, J. F. (1992). Extending and Formalizing the Framework for Information Systems Architecture. IBM Systems Journal, 31(3), 590 - 616.

Zave, P. (1991). An Insider's Evaluation of PAISLey. IEEE Transactions of Software Engineering, SE-17(3), 212 - 225.

Appendix A
A Notation to Reason About the Meaningfulness of Enterprise and Information Diagrams

A.1 An Introduction

The purpose of the operators presented in this Appendix is to facilitate the formal description and reasoning about the meaningfulness of enterprise and information diagrams. A meaningful diagram, is a diagram from which requirements can be derived, it is also a diagram that is pertinent and meaningful to both the problem owner and the problem solver. Thus it is by use of the enterprise and information diagrams that bridge is constructed from the problem domain to the solution domain, and it is by the use of this bridge that policy statements can be examined and their implications explored.

By using a well defined engineering language (such as Z) organisational goals, policies and statements can be expressed and validated with reference to the model. Consequently questions such as whether a particular model encapsulate a particular policy statement can be answered and reasoned about. The engineering language and its associated abstract syntax provides a medium of communication through which the problem owners can communicate and express their concerns. The engineering language also functions as an environment from within which the system can be reasoned about. In addition, the environment also functions in such a way so that requirements and their implications can be examined in a rigorous manner and conflicts, contradictions and omissions detected. As a direct result of the use of an engineering language both the problem solvers and problems owners can have confidence that they both understand the problem and its solution.

In this appendix the term actor is used to mean an agent in the enterprise projection and a party in the information projection. In addition the terms X, Y and Z are used to express the sets of all actors, roles and relationships respectively. In the following sections a set of operators will be specified in Z. Through the use of these operators meaningfulness and consistency can be examined.

A.2 The Tree Relation

Before we can begin to build up our operators will need to define what a tree structure is in Z. This is done in Figure A.2.1.

$$A \ddagger A == \{ f : A \pfun A \mid \exists_1 x : A \bullet x \in \operatorname{ran} f \land x \notin \operatorname{dom} f \land (\forall m : \operatorname{dom} f \bullet \exists i : \mathbb{N}_1 \bullet m \mapsto m \in f^{i}) \land (\forall q : \operatorname{dom} f ; n : \mathbb{N}_1 \bullet q \mapsto q \notin f^{n}) \}$$

Figure A.2.1 - The Tree Relation.

A.3 The Parent Function

The parent function allows us to define and examine the idea of parentage by using the idea of a tree structure. This function allows us to express the idea of lineage. The left hand side of the function is an ordered pair, the first argument of which is a nonnegative integer and the second argument is the offspring. The right hand side of the parent function represents the parentage. The meaning of this function is best understood with an illustration. The Z generic schema specification for this operator is given in Figure A.3.1.

$$[Q]$$

$$\text{Parent} : (\mathbb{N}_0 \times Q) \pfun \mathbb{P} Q$$

$$\forall T : Q \ddagger Q \bullet (\forall n : \mathbb{N}_0 ; x, y : X \mid x \mapsto y \in T^{n} \bullet \text{Parent}(n, x) = \{y\}) \land (\forall n : \mathbb{N}_0 ; x, y : X \mid \neg (x \mapsto y \in T^{n}) \bullet \text{Parent}(n, x) = \varnothing)$$

Figure A.3.1 The Parent Operator

The example given below is telling us that *B* is the grandparent of *A*. The length of lineage, grandparent, is denoted by the *2*.

$$\text{Parent}(2, A) = B$$

It is important to note that if the number of the left hand side of the ordered pair is zero then operator should return the right hand side of the order pair as is illustrated bellow.

$$Parent(0,A) = A$$

A.4 The Role to Role Operator

The purpose of the role to role operator is to allow us to express the idea that a role can be connected to another in the abstract syntax of the diagrams. The use of this operator is best illustrated by means of an example.

$$A \Upsilon B$$

The previous example tells us that the role A is connected to role B somehow, but it does not tell us how this connection is made. The Z axiomatic schema specification for this operator is given in Figure A.4.1

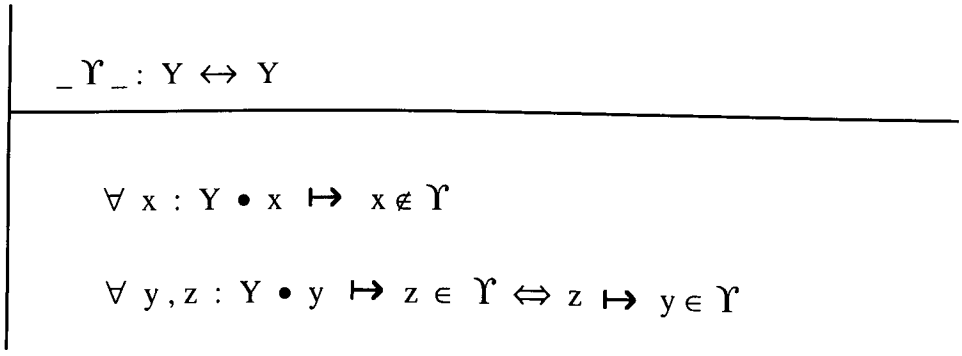


Figure A.4.1 - Role connect Role Operator.

A.5 The Actor Role Operator

The purpose of the actor role operator is to allow us to express the idea that an actor can contain a role of some description. The left hand side of the operator is the actor which will contain the role. The right hand side of the operator is the role contained with in that actor. The use of this operator is best illustrated by means of an example.

$$A \partial B$$

The above example tells us that the actor A contains the role B. The Z axiomatic schema specification for this operator is given in Figure A.5.1

$$\_ \partial \_ : X \leftrightarrow Y$$

$$\forall y : \text{ran } \partial \bullet \exists_1 x : \text{dom } \partial \bullet x \mapsto y \in \partial$$

Figure A.5.1 - Actor Role Operator.

A.6 The Actor Link Operator

The purpose of the actor link operator is to allow us to express the idea that an actor can have a link of some description connecting into it.. The left hand side of the operator is the agent which has the link connecting to it. The right hand side of the operator is the link that connects to the actor.

$$\_ \lambda \_ : X \leftrightarrow Z$$

$$\forall z : X ; \forall w, x, y : Z \mid z \mapsto w \in \lambda \wedge z \mapsto x \in \lambda \wedge z \mapsto y \in \lambda \bullet w = x \wedge w = y \wedge x = y$$

Figure A.6.1 - Actor Link Operator.

The use of this operator is best illustrated by means of an example.

$$A \lambda B$$

The above example tells us that the actor A has a link B which connects to it. The Z axiomatic schema specification for this operator is given in Figure A.6.1

A.7 The Role Link Operator

The role link operator allows us to express the idea that a link can connect to a role. The argument on the left hand side is the role and the argument on the right hand side is the link.

Let me now illustrate this operator with an example.

$$A \eta B$$

The above example tells us that role A is connected to link B. The axiomatic schema specification for this operator is given in Figure A.7.1

$$\_ \eta \_ : Y \leftrightarrow Z$$

$$\forall z : Y ; \forall w, x, y : Z \mid z \mapsto w \in \eta \wedge z \mapsto x \in \eta \wedge z \mapsto y \in \eta \bullet w = x \wedge w = y \wedge x = y$$

Figure A.7.1 Role Link Operator

A.8 The Actor SubType Type Operator

A.8.1 The Actor SubType Type Role Operator

The actor subtype type role operator allows us to express the idea that a role, which is contained in an actor is decomposed. The left hand side of the operator consists of an ordered pair. The first argument in the ordered pair is the actor that contain the role. The second argument is the role. The right hand side of the operator is the supertype of the role.

$$\_ \nabla_{role} \_ : (X \times Y) \nrightarrow Y$$

$$\forall a : X ; b : Y ; d : Y \bullet a \mapsto b \in \text{dom} \nabla_{role} \wedge d \in \text{ran} \nabla_{role} \Leftrightarrow \exists c : X \bullet \text{Parent}(1, a) = \{c\} \wedge \text{Parent}(1, b) = \{d\} \wedge a \mapsto b \in \partial \wedge d \mapsto c \in \partial$$

Figure A.8.1.1 - Actor Subtype Type Role Operator

This operator is best illustrated by an example.

$$(A, B) \nabla_{role} C$$

This example tells that a actor A contains a role B which is descended from role C. The Z axiomatic schema specification for this operator is given in Figure A.8.1.1

A.8.2 The Actor SubType Type Link Operator

The actor subtype type link operator allows us to express the idea that a link which is connected in an actor is decomposed. The left hand side of the operator consists of an ordered pair. The first argument in the ordered pair is the actor that the link connects to. The second argument is the link. The right hand side of the operator is the supertype of the link.

$$\_ \nabla_{link} \_ : (X \times Z) \leftrightarrow Z$$

$$\forall a : X;\ b : Z;\ d : Z \bullet$$
$$a \mapsto b \in \mathrm{dom}\, \nabla_{link} \wedge d \in \mathrm{ran}\, \nabla_{link} \Leftrightarrow$$
$$\exists c : X \bullet \mathrm{Parent}(1, a) = \{c\} \wedge \mathrm{Parent}(1, b) = \{d\} \wedge$$
$$a \mapsto b \in \lambda \wedge d \mapsto c \in \lambda$$

Figure A.8.2.1 - Actor Subtype Type Link Operator

This operator is best illustrated by an example.

$$(A, B) \nabla_{link} C$$

The above example tells that a actor A which has a link B connected to it is descended from link C. The Z axiomatic schema specification for this operator is given in Figure A.8.2.1

A.9 The Actor Role Link Operator

This operator allows us to express the idea that an actor which contains a role can have a link connected to that role. The left hand side of this operator consists of an ordered pair. The first argument is the actor which contains the role, the second argument is the role itself. The right hand side of the operator is the

link which connects to the role. Let us now illustrate this operator with an example.

$$(A, B) \Diamond C$$

The above example tells us that an actor A contains a role B that has a link C connecting into it. The Z axiomatic schema specification for this operator is given in Figure A9.1

$$\_ \Diamond \_ : (X \times Y) \leftrightarrow Z$$

$$\forall x : X ; y : Y ; z : Z \bullet$$

$$(x, y) \mapsto z \in \Diamond \Leftrightarrow$$

$$x \mapsto y \in \partial \wedge y \mapsto z \in \Diamond \wedge x \mapsto z \in \lambda$$

Figure A.9.1 - Actor Role Link Operator

A.10 The Actor Type Number Operator

A.10.1 The Actor Type Number Role Operator

This operator allows us to express the idea that an actor can contain a specific number of a specific type of roles. The left hand side of this operator is an ordered pair, the first argument of the which is an actor, and the second argument of which is a type of the role. The right hand side of the operator is the number of times that a role of that type is contained in the actor.

$$\_ \vartheta_{role} \_ : (X \times Y) \nrightarrow N_0$$

$$\forall x : X ; a : Y ; n : N_0 \bullet (x, a) \vartheta_{role} n \Leftrightarrow$$

$$\#\{ z : Y \mid x \partial z \wedge Parent(1, z) = \{ a \} \wedge \exists w : X \bullet$$

$$Parent(1, x) = \{ w \} \wedge w \partial a \} = n$$

Figure A.10.1.1 Actor Type Number Role Operator

Let me now illustrate this operator with an example.

$$(A , B) \vartheta_{role} C$$

The above example tells us that actor A contains C in number roles of type B. The Z axiomatic schema specification for this operator is given in Figure A.10.1.1

A.10.2 The Actor Type Number Link Operator

This operator allows us to express the idea that an actor can be connected to a specific number of a specific type of links. The left hand side of this operator is an ordered pair. The first argument is an actor while the second argument is a type of the link that connects to that actor. The right hand side of the operator is the number of times that the link of that type is connected to the actor.

Let me now illustrate this operator with an example.

$$(A , B) \vartheta_{link} C$$

The above example tells us that actor A is connected to C in a number links of type B. The Z axiomatic schema specification for this operator is given in Figure A10.2.1

$$- \vartheta_{link} - : (X \times Z) \pfun N_0$$

$$\forall x : X ; a : Z ; n : N_0 \bullet (x , a) \vartheta_{link} n \Leftrightarrow$$

$$\# \{ z : Z \mid x \lambda z \land Parent(1 , z) = \{ a \} \land \exists w : X \bullet$$

$$Parent(1 , x) = \{ w \} \land w \lambda a \} = n$$

Figure A.10.2.1 Actor Type Number Link Operator

Appendix B
Specifying and Reasoning about the Meaningfulness of Enterprise and Information Diagrams

B.1 Introduction

The purpose of specifying and verifying organisational rules is to identify where possible conflicts might exist within an organisation. Organisational rules are rules that govern what is deemed to be a meaningful enterprise and information diagram. In section B.2 the basic sets of objects and relationships that can exist within these diagrams are described. Then in section B.3 the rules governing what will be deemed a meaningful diagram are presented. A meaningful diagram is a diagram which correctly expresses the organisation and thus acts as a bridge between the problem domain and the solution domain.

B.2 The Basic Sets

The idea of parentage can be modelled by using a mapping. So we may say that an object can map to its parent. This gives rise to the need for a base set of types for each type of object that can exist in a role relation diagram. For the enterprise projection he base set of type of links for instrumental interfaces and obligation relations will be called the *instrumental base set* and the *obligation base set* respectively. In addition the base set of types for the structural and functional roles will be called a *structural base set* and a *functional base set* respectively.

For the information projection the base set of type of links for conversational relationships and contractual relationships will be called the *conversational relation base set* and the *contractual relation base set* respectively. In addition the base set of types for the contractual and conversational roles will be called a *contractual base set* and a *conversational base set* respectively.

The following notation will be used to describe the sets of objects that exist within a role relation diagram. The base sets of objects are the sets of building blocks (types) from which a role relation diagram is constructed.

β_{ST} - is the base set of all structural roles.

β_{FR} - is the base set of all functional roles.

β_{II} - is the base set of all instrumental interfaces.

$\beta_{\acute{O}R}$ - is the base set of all obligation relations.

β_{CO} - is the base set of all contractual roles.

β_{CV} - is the base set of all conversational roles.

β_{CN} - is the base set of all conversational relationships.

β_{CT} - is the base set of all contractual relationships.

A small phi is used to denote the set of elements not in the base set, so that $\beta_x \cap \pi_x = \varnothing$ and this set is referred to as the phi set. For example, the union of the base set of structural roles and its $\beta_{ST} \cap \pi_{ST} = \varnothing$. A capital delta is used to denote the set of all elements such that the following holds: $\Delta_x = \beta_x \cup \pi_x$

Δ_{ST} - is the set of all structural roles.

Δ_{FR} - is the set of all functional roles.

Δ_{II} - is the set of all instrumental interfaces.

Δ_{OR} - is the set of all obligation relationships.

Δ_{CO} - is the set of all contractual roles.

Δ_{CV} - is the set of all conversational roles.

Δ_{CN} - is the set of all conversational relationships.

Δ_{CT} - is the set of all contractual relationships.

Within the enterprise projection the set of all the agent entities is denoted by the symbol X_E the set of all role is denoted by the symbol Y_E and the set of all links is denoted by the symbol Z_E.

X_E - is the set of all agent entities.

Y_E - is the set of all roles. i.e. $Y_E = \Delta_{ST} \cup \Delta_{FR}$

Z_E - is the set of all links. i.e. $Z_E = \Delta_{OR} \cup \Delta_{II}$

Within the information projection the set of all the parties is denoted by the symbol X_I the set of all role is denoted by the symbol Y_I, and the set of all links is denoted by the symbol Z_I.

X_I - is the set of all parties.

Y_I - is the set of all roles. i.e.. $Y_I = \Delta_{CO} \cup \Delta_{CV}$

Z_I - is the set of all links. i.e.. $Z_I = \Delta_{CT} \cup \Delta_{CN}$

B.3 Rules Governing Meaningfullness

In this section each rule will be presented along with a proposition that should hold for the entire enterprise diagram. It should be noted that it is possible for more rules to be added or for rules to be removed. It is the ability that it gives to the problem solvers and problem owners to insert and delete rules that make the formal notation so flexible and hence so expressive.

B.3.1 Rule One

B.3.1.1 The Enterprise Rule

Every Agent Entity in an Enterprise Diagram contains one or more Structural Roles

B.3.1.2 The Enterprise Proposition

$Rule_1 \Leftrightarrow \forall x{:}X_E \; ; \exists s{:}\Delta_{ST} \bullet x \partial s$

B.3.1.3 The Information Rule

Every Party in an Information Diagram contains one or more Contractual Roles

B.3.1.4 The Information Proposition

$$Rule_1 \Leftrightarrow \forall x{:}X_I \; ; \exists s{:}\Delta_{CO} \bullet x \partial s$$

B.3.2 Rule Two

B.3.2.1 The Enterprise Rule

Every Role in an Enterpise Diagram is contained in exactly one Agent Entity.

B.3.2.2 The Enterprise Proposition

$$Rule_2 \Leftrightarrow \forall y{:}Y_E ; \exists_1 x{:}X_E \bullet x \partial y$$

B.3.2.3 The Information Rule

Every Role in an Information Diagram is contained in exactly one Party.

B.3.2.4 The Information Proposition

$$Rule_2 \Leftrightarrow \forall y{:}Y_I ; \exists_1 x{:}X_I \bullet x \partial y$$

B.3.3 Rule Three

B.3.3.1 The Enterprise Rule

The only Agent Entity without a parent is called *The_World.*

And

Every Agent Entity has exactly one parent with the exception of *The_World* agent.

And

Form *The_World* agent entity it is possible to get to every other Agent Entity via the Parent Relation.

And

Every Parent Agent Entity has two or more children.

B.3.3.2 The Enterprise Proposition

$$\begin{aligned} Rule_3 \Leftrightarrow\ & \exists x{:}X_E \bullet (\forall a{:}X_E \bullet \neg parent(1,x) = a \wedge x = The\_World \wedge \\ & \forall z{:}X_E | \forall y{:}X_E \bullet \neg parent(1,z) = y \bullet y = x) \\ & \forall a{:}X_E | x \neq The\_World \bullet \exists_1 b{:}X_E \bullet parent(1,b) = a \\ & \forall c{:}X_E ; \exists n{:}\aleph_0 \bullet parent(n,c) = The\_World \\ & \forall x{:}X_E | (\forall z{:}X_E \bullet \neg parent(1,y) = x) \bullet \exists z,y{:}X_E \bullet y \neq z \wedge \\ & parent(1,y) = x \wedge parent(1,z) = x \end{aligned}$$

B.3.3.3 The Information Rule

The only Party without a parent is called *The_World.*

And

Every Party has exactly one parent with the exception of *The_World* agent.

And

Form *The_World* party it is possible to get to every other Party via the Parent Relation.

And

Every Parent Party has two or more children.

B.3.3.4 The Information Proposition

$$\begin{aligned} Rule_3 \Leftrightarrow\ & \exists x{:}X_I \bullet (\forall a{:}X_I \bullet \neg parent(1,x) = a \wedge x = The\_World \wedge \\ & \forall z{:}X_I | \forall y{:}X_E \bullet \neg parent(1,z) = y \bullet y = x) \\ & \forall a{:}X_I | x \neq The\_World \bullet \exists_1 b{:}X_I \bullet parent(1,b) = a \\ & \forall c{:}X_I ; \exists n{:}\aleph_0 \bullet parent(n,c) = The\_World \\ & \forall x{:}X_I | (\forall z{:}X_I \bullet \neg parent(1,y) = x) \bullet \exists z,y{:}X_I \bullet y \neq z \wedge \\ & parent(1,y) = x \wedge parent(1,z) = x \end{aligned}$$

B.3.4 Rule Four

B.3.4.1 The Enterprise Rule

There are no Obligation Relations that connect to Functional Roles.

And

There are no Instrumental Interfaces that connect to Structural Roles.

And

Every link is either a child or a member of the base set.

B.3.4.2 The Enterprise Proposition

$$\begin{aligned} Rule_4 \Leftrightarrow\ & \forall x{:}\Delta_{OR};\forall y{:}\Delta_{FR} \bullet \neg y\,\eta\,x \\ & \forall x{:}\Delta_{II};\forall y{:}\Delta_{ST} \bullet \neg y\,\eta\,x \\ & \forall x{:}Z_E;\exists y{:}(\beta_{OR}\cup\beta_{II}) \bullet \\ & \quad parent(1,y)=x \vee x\in(\beta_{OR}\cup\beta_{II}) \wedge \\ & \quad \neg(parent(1,y)=x \wedge x\in(\beta_{OR}\cup\beta_{II})) \end{aligned}$$

B.3.4.3 The Information Rule

There are no Contractual Relationships that connect to Conversational Roles.

And

There are no Conversational Relationships that connect to Contractual Roles.

And

Every link is either a child or a member of the base set.

B.3.4.4 The Information Proposition

$$\begin{aligned} Rule_4 \Leftrightarrow\ & \forall x{:}\Delta_{CT};\forall y{:}\Delta_{CV} \bullet \neg y\,\eta\,x \\ & \forall x{:}\Delta_{CN};\forall y{:}\Delta_{CO} \bullet \neg y\,\eta\,x \\ & \forall x{:}Z_I;\exists y{:}(\beta_{CT}\cup\beta_{CN}) \bullet \\ & \quad parent(1,y)=x \vee x\in(\beta_{CT}\cup\beta_{CN}) \wedge \\ & \quad \neg(parent(1,y)=x \wedge x\in(\beta_{CT}\cup\beta_{CN})) \end{aligned}$$

B.3.5 Rule Five

B.3.5.1 The Enterprise Rule

Every link connects together exactly two roles in different agent entities and it is true that the different agents share the same parent at some number greater that zero.

B.3.5.2 The Enterprise Proposition

$$Rule_5 \Leftrightarrow \forall z : Z_E \; ; \exists a,b : Y_E \; ; \exists c,d : X_E \bullet$$
$$c \partial a \wedge d \partial b \wedge a \eta z \wedge b \eta z \wedge c \neq d \wedge c \lambda z \wedge d \lambda z \wedge$$
$$(\forall w : Y_E \bullet w \eta z \Rightarrow (w = a \vee w = b)) \wedge$$
$$(\exists n_1, n_2 : \aleph_1 \bullet parent(n_1, c) = parent(n_2, d))$$

B.3.5.3 The Information Rule

Every link connects together exactly two roles in different party entities and it is true that the different parties share the same parent at some number greater that zero.

B.3.5.4 The Information Proposition

$$Rule_5 \Leftrightarrow \forall z : Z_I \; ; \exists a,b : Y_I \; ; \exists c,d : X_I \bullet$$
$$c \partial a \wedge d \partial b \wedge a \eta z \wedge b \eta z \wedge c \neq d \wedge c \lambda z \wedge d \lambda z \wedge$$
$$(\forall w : Y_I \bullet w \eta z \Rightarrow (w = a \vee w = b)) \wedge$$
$$(\exists n_1, n_2 : \aleph_1 \bullet parent(n_1, c) = parent(n_2, d))$$

B.3.6 Rule Six

B.3.6.1 The Enterprise Rule

For every agent entity that has children the roles contained in the child are either members of the base set of roles or else children of roles contained in the parent agent.

And

For every agent entity that has children there is at least one role contained in the children such that it is a child of a role contained in its parent.

B.3.6.2 The Enterprise Proposition

$$
\begin{aligned}
Rule_6 \Leftrightarrow \quad & \forall a{:}X_E \,|\, (\exists b{:}X_E \bullet parent(1,b) = a) \bullet \\
& \forall r{:}\{q{:}Y_E \bullet a \partial r\}\,; \\
& \forall s{:}\,\{t{:}Y_E \bullet w \in X_E \wedge parent(1,w) = a \wedge w \partial t\} \bullet \\
& s \in (\beta_{ST} \cup \beta_{FR}) \vee parent(1,s) = r \wedge \\
& \neg(s \in (\beta_{ST} \cup \beta_{FR}) \wedge parent(1,s) = r) \\
\forall a{:}X_E \,|\, & (\exists b{:}X_E \bullet parent(1,b) = a) \bullet \\
& \exists i : X_E\,; \exists f{:}Y_E\,; \exists e : Y_E \bullet \\
& parent(1,c) = a \wedge a \partial f \wedge c \partial e \wedge parent(1,e) = f
\end{aligned}
$$

B.3.6.3 The Information Rule

For every party that has children the roles contained in the child are either members of the base set of roles or else children of roles contained in the parent party.

And

For every party that has children there is at least one role contained in the children such that it is a child of a role contained in its parent.

B.3.6.4 The Information Proposition

$$
\begin{aligned}
Rule_6 \Leftrightarrow \quad & \forall a{:}X_I \,|\, (\exists b{:}X_I \bullet parent(1,b) = a) \bullet \\
& \forall r{:}\{q{:}Y_I \bullet a \partial r\}\,; \\
& \forall s{:}\,\{t{:}Y_I \bullet w \in X_E \wedge parent(1,w) = a \wedge w \partial t\} \bullet \\
& s \in (\beta_{CO} \cup \beta_{CV}) \vee parent(1,s) = r \wedge \\
& \neg(s \in (\beta_{CO} \cup \beta_{CV}) \wedge parent(1,s) = r) \\
\forall a{:}X_I \,|\, & (\exists b{:}X_I \bullet parent(1,b) = a) \bullet \\
& \exists i : X_I\,; \exists f{:}Y_I\,; \exists e : Y_I \bullet \\
& parent(1,c) = a \wedge a \partial f \wedge c \partial e \wedge parent(1,e) = f
\end{aligned}
$$

B.3.7 Rule Seven

B.3.7.1 The Enterprise Rule

For every link that is a parent it is true that the roles that connect to the parent link are parents of the roles that connect to the child of the parent link

And

For every link that is a parent it is true that the agents that are connected to the parent link are parents of the agents that are connected to the child of the parent link.

B.3.7.2 The Enterprise Proposition

$$
\begin{aligned}
Rule_7 \Leftrightarrow\ & \forall a{:}Z_E \,|\, (\exists b{:}Z_E \bullet parent(1,b) = a) \bullet \\
& \forall r{:}Y_E \bullet a\,\eta\, r \ \Rightarrow\ \exists x{:}Z_E ; \exists y{:}Y_E \bullet \\
& parent(1,x) = a \wedge parent(1,y) = r \wedge x\,\eta\, y \\
& \forall a{:}Z_E \,|\, (\exists b{:}Z_E \bullet parent(1,b) = a) \bullet \\
& \forall r{:}X_E \bullet a\,\lambda\, r \ \Rightarrow\ \exists x{:}Z_E ; \exists y{:}X_E \bullet \\
& parent(1,x) = a \wedge parent(1,y) = r \wedge x\,\lambda\, y
\end{aligned}
$$

B.3.7.3 The Information Rule

For every link that is a parent it is true that the roles that connect to the parent link are parents of the roles that connect to the child of the parent link

And

For every link that is a parent it is true that the parties that are connected to the parent link are parents of the parties that are connected to the child of the parent link.

B.3.7.4 The Information Proposition

$$
\begin{aligned}
Rule_7 \Leftrightarrow\ & \forall a{:}Z_I \,|\, (\exists b{:}Z_I \bullet parent(1,b) = a) \bullet \\
& \quad \forall r{:}Y_I \bullet a\,\eta\, r \ \Rightarrow\ \exists x{:}Z_I ; \exists y{:}Y_I \bullet \\
& \quad parent(1,x) = a \wedge parent(1,y) = r \wedge x\,\eta\, y \\
& \forall a{:}Z_I \,|\, (\exists b{:}Z_I \bullet parent(1,b) = a) \bullet \\
& \quad \forall r{:}X_I \bullet a\,\lambda\, r \ \Rightarrow\ \exists x{:}Z_I ; \exists y{:}X_I \bullet \\
& \quad parent(1,x) = a \wedge parent(1,y) = r \wedge x\,\lambda\, y
\end{aligned}
$$

Appendix C
A Notation to Reason About the Meaningfulness of Interaction and Conversation Diagrams

C.1 An Introduction

The purpose of the operators and set definitions presented in this Appendix is to facilitate the formal description and reasoning about the meaningfulness of interaction and conversation diagrams. A meaningful diagram is a diagram from which requirements can be derived, and is also a diagram that is pertinent and meaningful to both the problem owner's and the problem solver's perception of the problem. In addition, it is one from which policy statements may be seen as emergent properties of the diagrams. Thus it is by use of the interaction and conversation diagrams that a bridge can be constructed from the problem domain to the solution domain, and it is by the use of this bridge that policy and requirements statements can be examined and their implications explored. From this exploration and examination confidence between the problem owners and problem solvers can be established.

In this appendix the term 'interaction' is used to express the linguistic behavioural model that is used in the enterprise projection. The term 'conversation' is used to express the linguistic behavioural model that is used in the information projection. In addition, the term 'actor' is used to denote the set of all entities that can directly manipulate the system state.

C.2 The Basic Sets and their Relationships

In order for us to specify and analyse some form of conversation or interaction, a formal framework is required. This framework takes the form of a set of basic set elements and a set of formal axiomatic relationships. The following notation will be used to describe the basic sets that comprise an interaction or conversation diagram.

Δ_{AC} - is the set of all actors.

Δ_{AT} - is the set of all speech types

Δ_{DT} - is the set of all decision types

Δ_{IT} - is the set of all instrumental types

Δ_{AM} - is the set of all projection attributes and their manipulations

Δ_{OM} - is the set of all projection objects and their manipulations

Δ_{ST} - is the set of all start acts.

Δ_{SP} - is the set of all stop acts.

Δ_{IA} - is the set of all instrumental acts.

Δ_{DA} - is the set of all decision acts.

Δ_{SA} - is the set of all speech acts.

Δ_{CC} - is the set of all conditionals.

It should be noted that the t'erm speech act type' is used to express what is speech act theory would be called a illocutionary act. However for the purpose of the formal framework the term speech act is used to describe a larger unit of interaction. Consequently the term 'speech act type' is used to express and denote the nature and force of the speech act. The term 'decision act type' is used to describe the nature of the decision act. In addition, the term 'instrumental act type' is used to describe the nature of the instrumental act.

The following are a set of definitions concerning how the sets relate to one another.

$$\Delta_{Object} = \Delta_{ST} \cup \Delta_{SP} \cup \Delta_{IA} \cup \Delta_{DA} \cup \Delta_{SA}$$

The above statement defines a type (or set) hierarchy, thus Δ_{Object} is the supertype and the sets on the right hand side of the = operator its sub type.

$$\varnothing = \Delta_{ST} \cap \Delta_{SP} = \Delta_{ST} \cap \Delta_{IA} = \Delta_{ST} \cap \Delta_{DA} = \Delta_{ST} \cap \Delta_{SA} \cup \Delta_{CC}$$

$$\varnothing = \Delta_{SP} \cap \Delta_{ST} = \Delta_{SP} \cap \Delta_{IA} = \Delta_{SP} \cap \Delta_{DA} = \Delta_{PT} \cap \Delta_{SA} \cup \Delta_{CC}$$

$$\varnothing = \Delta_{IA} \cap \Delta_{ST} = \Delta_{IA} \cap \Delta_{SP} = \Delta_{IA} \cap \Delta_{DA} = \Delta_{SA} \cap \Delta_{SA} \cup \Delta_{CC}$$

$$\varnothing = \Delta_{DA} \cap \Delta_{ST} = \Delta_{DA} \cap \Delta_{SP} = \Delta_{DA} \cap \Delta_{IA} = \Delta_{DA} \cap \Delta_{SA} \cup \Delta_{CC}$$

$$\varnothing = \Delta_{SA} \cap \Delta_{ST} = \Delta_{SA} \cap \Delta_{SP} = \Delta_{SA} \cap \Delta_{IA} = \Delta_{SA} \cap \Delta_{DA} \cup \Delta_{CC}$$

The above statements confirm the axiom that every set is unique with reference to the other sets.

C.3 Specifying the Basic Sets and their Relationships in Z

In this section the basic sets and their relationships will be specified in Z. The purpose is specifying them in Z is to provide a formal conceptual platform from which a set of functions may be derived and reasoned about. The goal of the functions is to provide a formal framework from with which policy and requirement statements may be modelled and elueidated.

$$[\text{Object}]$$
$$[\Delta_{AT}, \Delta_{DT}, \Delta_{IT}]$$
$$[\Delta_{AC}, \Delta_{OM}, \Delta_{AM}]$$
$$[\Delta_{CC}]$$

Figure C.3.1 The Parachuted Types

In Figure C.3.1 the types that will be used to construct and reason about the formal framework of interactions and conversations are depicted. These type definitions begin with the parachuting in of the supertype. This will form the set from which all of the other sets of objects in the model will be derived. The next line defines the attributes which will be ascribed, via a function, to some of the objects which are derived from the supertype. The final line defines the attributes which will be attributed, via a function, to all of the objects which are descended from the supertype.

$$\Delta_{\text{Object}} : \mathbf{P}\ \text{Object}$$

Figure C.3.2 The Set Containing All Objects

In C.3.2 a set is defined which will be the union of all the basic object sets that will exists within our model. The above diagram defines a set called $\Delta_{Objects}$ to be an element in the power set of *Object*.

In Figure C.3.3 the basic sets and their relationships are depicted in an axiomatic schema. In the declaration part of the schema the basic sets are defined to be of type power set of Object. Then in the axiom part of the schema the set of relationships that exists between the sets are defined. There are two points that should be noted about the basic sets and their relationships. Firstly, all of the basic sets are defined to be subsets of the set of all objects. Secondly, the intersection of each of the

base sets with its fellows is defined to be the empty set. This statement embodies the assertion that each element of the base sets are unique and exists in only one set.

$$\Delta_{ST} : \mathbf{P}\ \text{Object}$$
$$\Delta_{SA} : \mathbf{P}\ \text{Object}$$
$$\Delta_{DA} : \mathbf{P}\ \text{Object}$$
$$\Delta_{IA} : \mathbf{P}\ \text{Object}$$
$$\Delta_{SP} : \mathbf{P}\ \text{Object}$$

$$\Delta_{ST} \subset \Delta_{Object}$$
$$\Delta_{SA} \subseteq \Delta_{Object}$$
$$\Delta_{DA} \subset \Delta_{Object}$$
$$\Delta_{IA} \subset \Delta_{Object}$$
$$\Delta_{SP} \subset \Delta_{Object}$$

$$\Delta_{ST} \cap \Delta_{SA} = \Delta_{ST} \cap \Delta_{DA} = \Delta_{ST} \cap \Delta_{IA} = \Delta_{ST} \cap \Delta_{SP} = \Delta_{ST} \cap \Delta_{CC} = \varnothing$$

$$\Delta_{SA} \cap \Delta_{ST} = \Delta_{SA} \cap \Delta_{DA} = \Delta_{SA} \cap \Delta_{IA} = \Delta_{SA} \cap \Delta_{SP} = \Delta_{SA} \cap \Delta_{CC} = \varnothing$$

$$\Delta_{DA} \cap \Delta_{ST} = \Delta_{DA} \cap \Delta_{SA} = \Delta_{DA} \cap \Delta_{IA} = \Delta_{DA} \cap \Delta_{SP} = \Delta_{DA} \cap \Delta_{CC} = \varnothing$$

$$\Delta_{IA} \cap \Delta_{ST} = \Delta_{IA} \cap \Delta_{SP} = \Delta_{IA} \cap \Delta_{DA} = \Delta_{IA} \cap \Delta_{SP} = \Delta_{IA} \cap \Delta_{CC} = \varnothing$$

$$\Delta_{SP} \cap \Delta_{ST} = \Delta_{SP} \cap \Delta_{SA} = \Delta_{SP} \cap \Delta_{DA} = \Delta_{SP} \cap \Delta_{IA} = \Delta_{SP} \cap \Delta_{CC} = \varnothing$$

$$\forall a : \Delta_{Object} \bullet$$
$$a \in \Delta_{SP} \vee a \in \Delta_{SA} \vee a \in \Delta_{DA} \vee$$
$$a \in \Delta_{IA} \vee a \in \Delta_{SP}$$

Figure C.3.3 The Base Sets and their Relationships

Finally, in Figure C.3.3 the last statement embodies the assertion that the superset called Δ_{Object} is the union of all five of the base sets. Thus every element in the set Δ_{Object} is also a member of one, and only one, of the base sets.

C.4 Specifying the Operators in Z

In this section the operators that will be used to specify and analyse an interaction or conversation diagram will be presented and described. The purpose of these operators is to provide a formal framework within which it is possible to examine policy and requirement statement and explore their impact on the system. In addition, these operators all facilitate the examination and classification of the emergent properties of the diagrams.

C.4.1 The Link Operator

Figure C.4.1.1 depicts the link operator that is used to connect together objects contained within an interaction or conversation diagram. The declaration part of the schema states that the function is a relation binding together two objects. Thus the input to the operator is an object and the output is an object.

$$\_\ \text{Link}\ \_ : \Delta_{\text{Object}} \leftrightarrow \Delta_{\text{Object}}$$

$$\forall a : \text{dom}\ (\text{Link}) \bullet$$
$$a \in \Delta_{ST} \vee a \in \Delta_{SA} \vee a \in \Delta_{DA} \vee a \in \Delta_{IA}$$

$$\forall a : \text{ran}\ (\text{Link}) \bullet$$
$$a \in \Delta_{SA} \vee a \in \Delta_{DA} \vee a \in \Delta_{IA} \vee a \in \Delta_{SP}$$

Figure C.4.1.1The Link Operator

The axiom part of the schema definition defines the limitation of the function. The first clause asserts that only objects of certain types may act as the input to the function, while the second clause asserts that only objects of certain types may act as the result of the function.

C.4.2 The Executor Operator

The executor operator is depicted in Figure C.4.2.1. This operator allows for the expression and analysis of which actor performs an element in the interaction or conversation diagram. Thus if speech acts, decision acts and instrumental acts are viewed as actions then we may ask the question "Who will perform this action ?". As a direct result we may formalise the policy and requirement expressions "only the pilot may make this decision" or, "only my supervisor can fire me".

$$\_\ \text{Executor}\ \_ : \Delta_{\text{Object}} \leftrightarrow \Delta_{AC}$$

$$\forall a : \text{dom}\ (\text{Executor}) \bullet$$
$$a \in \Delta_{SA} \vee a \in \Delta_{DA} \vee a \in \Delta_{IA}$$

$$\forall a : (\Delta_{SA} \cup \Delta_{DA} \cup \Delta_{IA}) \bullet$$
$$\exists_1 b : \Delta_{AC} \bullet a\ \text{Executor}\ b$$

Figure C.4.2.1 The Executor Operator.

In Figure C.4.2.1 the executor operator is illustrated and defined. In the declaration part of the axiomatic schema definition the operator is defined to have one input of type Δ_{Object} and to produce one output of type Δ_{AC}. In the axiom part of the

schema definition two assertions are made about the operator. The first is that the only objects upon which we may use this operator are speech acts, decision acts and instrumental acts, while the second is that each input to the operator produces exactly one output and that all of the objects of type speech act, decision act and instrumental act produce an actor as their output.

C.4.3 The Utterance Operator

The utterance operator is depicted in Figure C.4.3.1. This operator allows for the expression and analysis of which speech act contains an utterance of this particular type and force in an interaction or conversation diagram. Thus if speech acts are viewed as actions then we may formally ask the question "What is the utterance associated with this action"? For example, we could precisely ask and answer the question "When in this set of diagrams is the Job Accept utterance made"?

In Figure C.4.3.1 the utterance operator is depicted. The operator is defined to have one input of type Δ_{Object} and to produce a single output of type Δ_{AT} in the declaration part of the axiomatic schema definition. In the axiom part of the schema definition two assertion are made about this operator. The first is that the only type of object upon which we may use this operator are speech acts (Δ_{SA}), while the second is that each input to this operator produces exactly one output and that all of the objects of type speech act produce an output.

$$\_ \text{ Utterance } \_ : \Delta_{Object} \leftrightarrow \Delta_{AT}$$

$$\forall a : \text{dom } (\text{Utterance }) \bullet a \in \Delta_{SA}$$

$$\forall a : \Delta_{SA} \bullet$$
$$\exists_1 b : \Delta_{AT} \bullet a \text{ Utterance } b$$

Figure C.4.3 The Utterance Operator

C.4.4 The Decision Operator

The decision operator is defined in Figure C.4.4.1. This operator allows for the expression and analysis of which decision act contains a decision of a particular type in an interaction or conversation diagram. Thus if decision acts are viewed as actions then we may formally ask the question "Is a decision of this kind ever made within the model?"

In Figure C.4.4.1 the decision operator is outlined. In the declaration part of the axiomatic schema definition the operator is defined to have one input of type

Δ_{Object} and to have an output of type Δ_{DT}. In the axiom part of the schema definition two assertions are made about this operator. The first is that the only type of object upon which we may use this operator is a decision act (Δ_{DA}), while the second is that each input to this operator produces exactly one output and that all of the objects of type decision act lie within the domain of the function.

$$\_ \text{ Decision } \_ : \Delta_{Object} \leftrightarrow \Delta_{DT}$$

$$\forall a : \text{dom } (\text{Decision }) \bullet a \in \Delta_{DA}$$

$$\forall a : \Delta_{DA} \bullet$$
$$\exists_1 b : \Delta_{DT} \bullet a \text{ Decision } b$$

Figure C.4.4.1 The Decision Operator

C.4.5 The Task Operator

The task operator is depicted in Figure C.4.5.1. This operator allows for the expression and analysis of which instrumental act contains a task of this type in an interaction or conversation diagram. Thus if instrumental acts are viewed as actions then we may formally ask the question "Who will perform this action "? So for example, we may ask the question of the interaction diagrams depicted in the first case study (Chapter 7 - Section 7.2.5) "Is the householders problem ever fixed"?

$$\_ \text{ Task } \_ : \Delta_{Object} \leftrightarrow \Delta_{IT}$$

$$\forall a : \text{dom } (\text{Task }) \bullet a \in \Delta_{IA}$$

$$\forall a : \Delta_{IA} \bullet$$
$$\exists_1 b : \Delta_{IT} \bullet a \text{ Task } b$$

Figure C.4.5.1 The Task Operator

In Figure C.4.5.1 the task operator is defined. The operator is defined to have one input of type Δ_{Object} and to produce one output of type Δ_{IA} (Instrumental Type) in the declaration part of the axiomatic schema.

In the axiom part of the schema definition two assertions are made about this operator. The first is that the only type of object upon which the operator may be used an instrumental act, while the second is that each input to this operator produces exactly one output and that all of the objects of type instrumental act lie within the domain of the function.

C.4.6 The Projection Object Operator

This operator allows for the expression and analysis of when a projection object is manipulated. Thus the questions like "Is this object every created ?" or "Where is this object manipulated ?" may be asked and answered within the formal framework of the operator. In Figure C.4.6.1 the projection object operator is depicted. In the declaration part of the axiomatic schema the operator is defined to have a single input of type Δ_{Object} and to produce a single output of type Δ_{OM}. The type Δ_{OM} is used to define the set of all projection objects and their manipulations.

$$\_ \text{Proj\_Obj} \_ : \Delta_{Object} \leftrightarrow \Delta_{OM}$$

$$\forall a : \text{dom}\ (\text{Proj\_Obj}) \bullet$$
$$a \in \Delta_{SP} \vee a \in \Delta_{DA} \vee a \in \Delta_{IA}$$
$$\forall a : (\Delta_{SP} \cup \Delta_{DA} \cup \Delta_{IA}) \bullet$$
$$\exists b : \Delta_{OM} \bullet a\ \text{Proj\_Obj}\ b$$
$$\forall a : \text{dom}\ (\text{Proj\_Obj}) \mid a\ \text{Proj\_Obj}\ \text{no\_access} \bullet$$
$$\nexists b : \text{ran}\ (\text{Proj\_Obj}) \bullet a\ \text{Proj\_Obj}\ b \wedge b \neq \text{no\_access}$$

Figure C.4.6.1 The Projection Object Operator

So for example, if there was just one projection object called *Report* and it only had two possible manipulations called *create* and *destroy*, then the set Δ_{OM} would be *{create(Report), destroy(Report), no_access}*. The element of the set called *no_access* is used to denote that no projection attributes are manipulated.

In the axiom part of the schema definition three assertion are made about this operator. The first is that the only types of objects upon which the this operator may be applied are speech acts, decision acts and instrumental acts. The second is that each input to this operator produces one or more outputs and that all of the objects of type speech act, decision act and instrumental act lie within the domain of the function. The third and final assertion is that if an input to the function produces *no_access* as its output then that is the only output that the function will produce for that given input.

C.4.7 The Projection Attribute Operator

This operator allows for the expression and analysis of when a projection attribute is manipulated in some way. Thus the questions like "Is this attribute ever created?" or "Where is this attribute manipulation performed" may be asked and answered within the formal framework of the operator.

In Figure C.4.7.1 the projection attribute operator is depicted. The operator is defined to have one input of type Δ_{Object} and to produce a single output of type Δ_{AM}. The type Δ_{AM} is used to define the set of all projection attributes and their manipulations. So for example, if there was just one projection object called *Data* and it only had two possible manipulations called *delete* and *access*, then the set Δ_{AM} would be *{delete(Data), access(Data), no_access}*. The element of the set called *no_access* is used to denote that no projection attributes are manipulated.

$$\_\ Proj\_Att\ \_ : \Delta_{Object} \leftrightarrow \Delta_{AM}$$

$$\forall a : dom\ (Proj\_Att) \bullet$$
$$a \in \Delta_{SP} \vee a \in \Delta_{DA} \vee a \in \Delta_{IA}$$
$$\forall a : (\Delta_{SP} \cup \Delta_{DA} \cup \Delta_{IA}) \bullet$$
$$\exists b : \Delta_{AM} \bullet a\ Proj\_Att\ b$$
$$\forall a : dom\ (Proj\_Att) \mid a\ Proj\_Att\ no\_access \bullet$$
$$\nexists b : ran\ (Proj\_Att) \bullet a\ Proj\_Att\ b \wedge b \neq no\_access$$

Figure C.4.7 The Projection Attribute Operator

In the axiom part of the schema definition three assertions are made about this operator. The first is that the only types of objects upon which the operator may be used are speech acts, decision acts and instrumental acts. The second is that each input to this operator produces one or more outputs and that all of the objects of type speech act, decision act and instrumental act lie within the domain of the function. The third and final assertion is that if an input to the function produces *no_access* as its output then that is the only output that the function will produce for that given input.

C.4.8 The Conditional Operator

Figure C.4.8.1 depicts the conditional operator that is used to extract the condition statements which are attached to links within an interaction or conversation diagram. The declaration part of the schema states that the function is a relation binding together the cartesian product of two objects with a set of conditionals. Thus the input to the operator is a cartesian product and the output is a set of conditionals.

The axiom part of the schema definition defines the limitations of the function. The first clause asserts that input to the function is a valid link between two objects in an interaction or conversation diagram. The second clause states that every link between two objects in an interaction or conversation diagram has a set of conditionals associated with it. The final clause states that every link has exactly one set of conditionals associated with it.

$$\_\ \text{Cond}\ \_ : (\Delta_{\text{Object}} \times \Delta_{\text{Object}}) \leftrightarrow \mathbf{P}\,\Delta_{CC}$$

$$\forall\ a,b : \Delta_{\text{Object}} \mid (a,b) \in \text{dom (Cond)} \bullet a\ \text{Link}\ b$$

$$\forall\ c,d : \Delta_{\text{Object}} \mid c\ \text{link}\ d \bullet (c,d) \in \text{dom (Cond)}$$

$$\forall\ e,f : \Delta_{\text{Object}} \mid (e,f) \in \text{dom (Cond)} \bullet$$

$$\exists_1\ g : \mathbf{P}\,\Delta_{CC} \bullet (e,f)\ \text{Cond}\ g$$

Figure C.4.1.1The Conditional Operator

C.4.9 The Pre Conditional Operator

Figure C.4.9.1 depicts the pre conditional operator that is used to extract the pre condition statements which are associated with nodes within an interaction or conversation diagram. The declaration part of the schema states that the function is a relation objects and conditionals.

$$\_\ \text{Pre}\ \_ : \Delta_{\text{Object}} \leftrightarrow \mathbf{P}\,\Delta_{CC}$$

$$\forall\ a : \Delta_{\text{Object}} \bullet a \in \text{dom (Pre)}$$

$$\forall\ b : \text{dom (Pre)} \bullet$$

$$\exists_1 c : \mathbf{P}\,\Delta_{CC} \bullet b\ \text{Pre}\ c$$

Figure C.4.1.1The Pre Conditional Operator

The axiom part of this schema definition defines the limitations of the function. The first clause asserts that every object in an interaction and conversation diagram has a set of pre conditionals associated with it. The second clause states that everything in the domain of the function has exactly one set of conditionals associated with it.

C.4.10 The Post Conditional Operator

Figure C.4.10.1 depicts the post conditional operator that is used to extract the post condition statements which are associated with nodes within an interaction or conversation diagram. The declaration part of the schema states that the function is a relation objects and conditionals.

The axiom part of this schema definition defines the limitations of the function. The first clause asserts that every object in an interaction and conversation

diagram has a set of post conditionals associated with it. The second clause states that everything in the domain of the function has exactly one set of conditionals associated with it.

$$\_\ \text{Post}\ \_ : \Delta_{\text{Object}} \leftrightarrow \mathbf{P}\,\Delta_{CC}$$

$$\forall\, a : \Delta_{\text{Object}} \bullet a \in \text{dom (Post)}$$

$$\forall\, b : \text{dom (Post)} \bullet$$

$$\exists_1 c : \mathbf{P}\,\Delta_{CC} \bullet b\ \text{Post}\ c$$

Figure C.4.10.1 The Post Conditional Operator

C.4.11 The Invarient Conditional Operator

Figure C.4.11.1 depicts the invarient conditional operator that is used to extract the invarient condition statements which are associated with nodes within an interaction or conversation diagram. The declaration part of the schema states that the function is a relation objects and conditionals.

$$\_\ \text{Inv}\ \_ : \Delta_{\text{Object}} \leftrightarrow \mathbf{P}\,\Delta_{CC}$$

$$\forall\, a : \Delta_{\text{Object}} \bullet a \in \text{dom (Inv)}$$

$$\forall\, b : \text{dom (Inv)} \bullet$$

$$\exists_1 c : \mathbf{P}\,\Delta_{CC} \bullet b\ \text{Inv}\ c$$

Figure C.4.10.1 The Post Conditional Operator

The axiom part of this schema definition defines the limitations of the function. The first clause asserts that every object in an interaction and conversation diagram has a set of invariant conditionals associated with it. The second clause states that everything in the domain of the function has exactly one set of conditionals associated with it.

C.5 Modelling Policy Statements

The purpose in providing a formal engineering language framework from within which to express and reason about conversation and interaction diagrams is firstly so that contradictions, omissions and errors can be detected and rectified. Thus questions such as "do any agents ever get into a position where they are responsible for *A* and also not responsible for *A*. Secondly it is to provide a means by which

organisational policy statements can be expressed and validated with reference to the model. As a result of these policy statements their implications and impact on the system can be described and represented back to the policy holder.

C.5.1 An Example of a Policy Statement

The policy statement which I will use for an example is taken from the world of system security and administration.

$Policy_1$ = Only the system administrator may write to the password file.

The above policy statement can be modelled at two levels, the first is the level of the actual object (password file) manipulation and the second is level of the speech act.

C.5.2 Modelling the Object Manipulation

The following is a formal expression of the policy statement stated in C.5.1 and modelled at the object manipulation level. At this level what is of concern is the life cycle of objects.

$$\forall a : \Delta_{Object} \mid a \; \mathrm{Pr}oj\_Obj\,(alter(passwd))$$
$$\bullet \; a \; Executor\,(system\_adm)$$

The above policy statement states that for every node in a conversation or interaction diagram which modifies the password file the only actor which can execute that node is the system administrator. The formal policy expression depicted above is a very simple interpretation of the original policy statement.

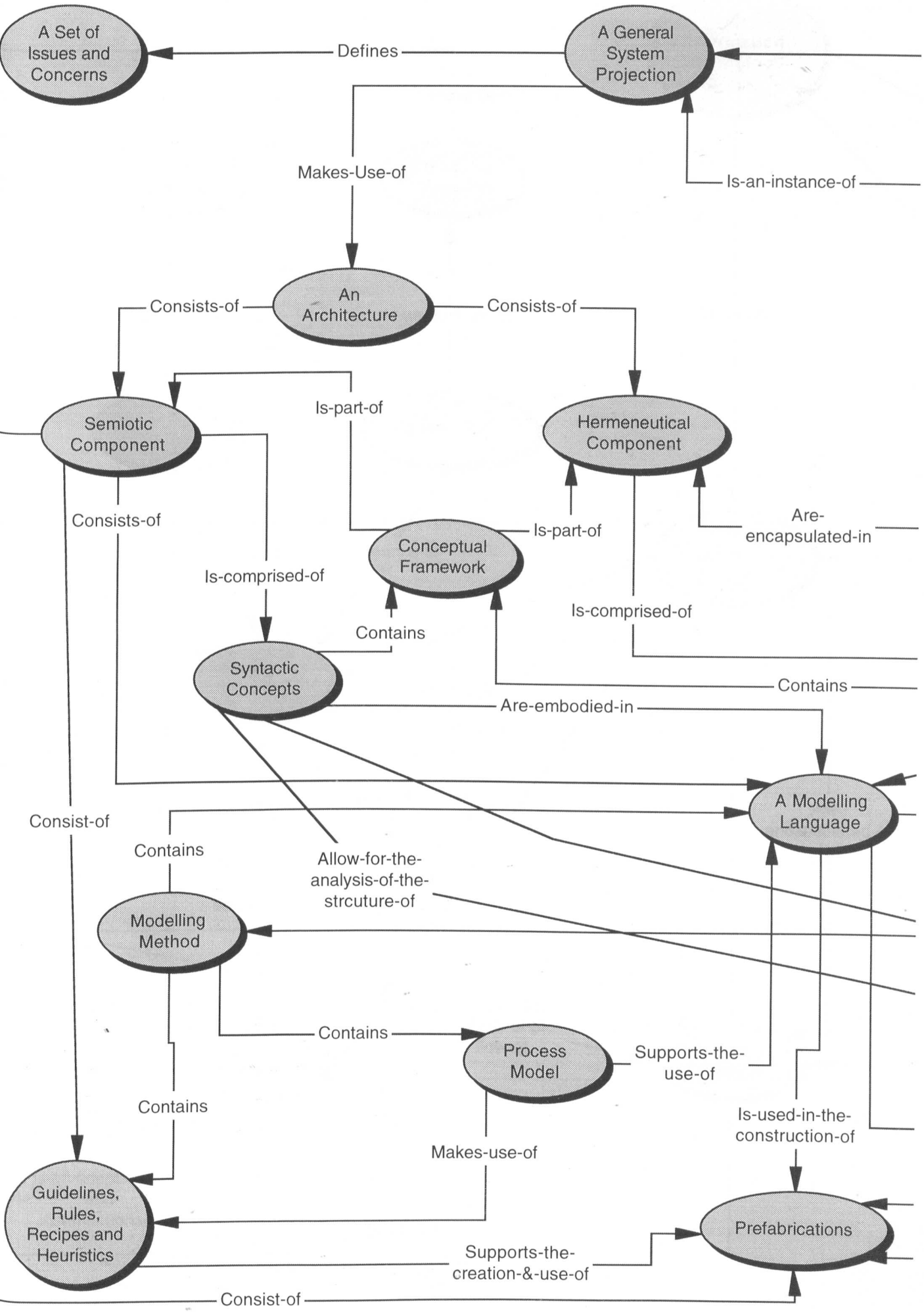
C.5.3 Modelling Speech Acts

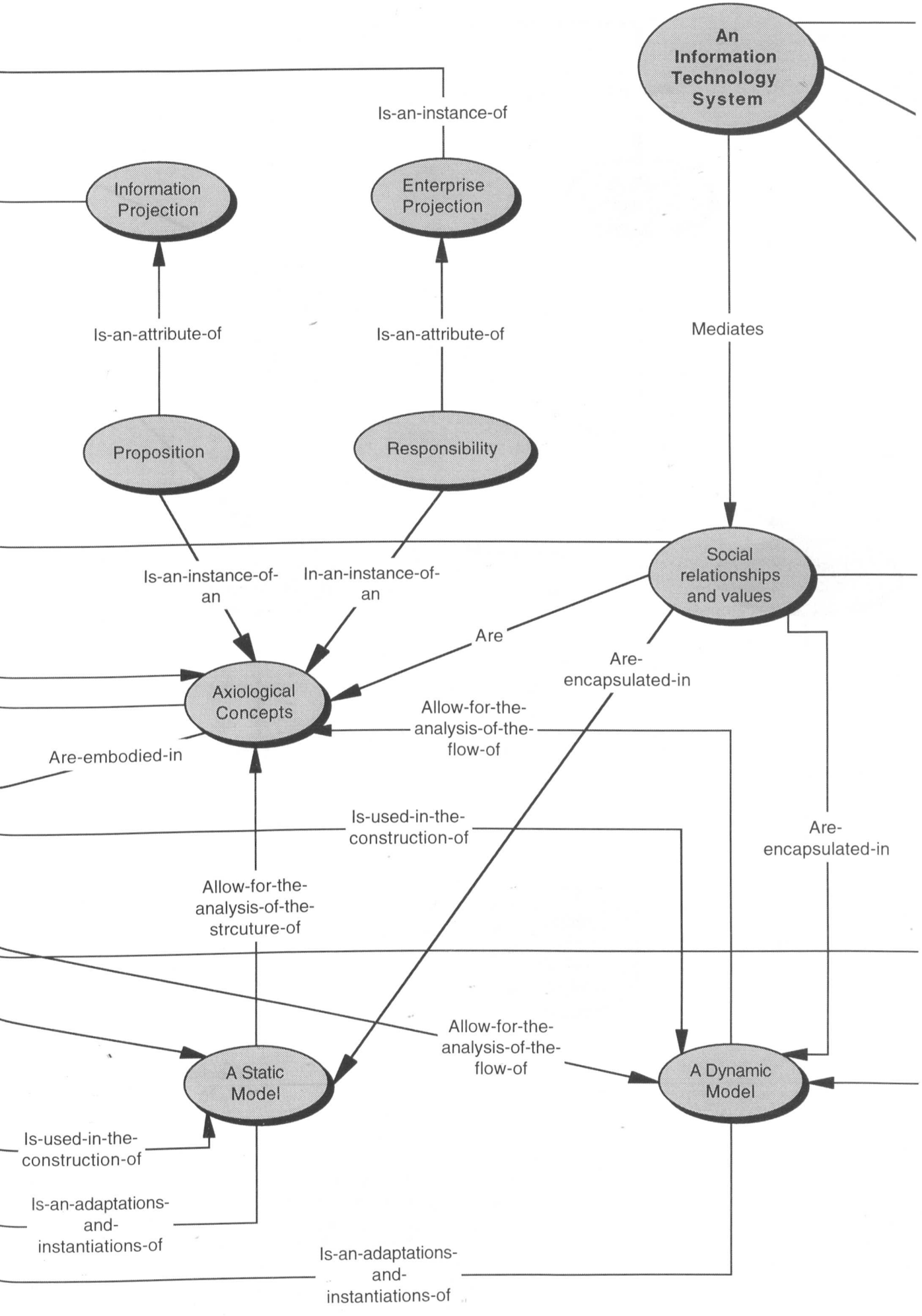
The following is a formal expression of the policy statement stated in C.5.1 and modelled at the speech act level. At this level what is of concern is the speech acts, and the intentions associated with them, made within the model. As with each speech act the question can be asked what is its purpose? The policy statement may thus be interpreted that only the system administrator may make utterances associated with the manipulation of the password file.

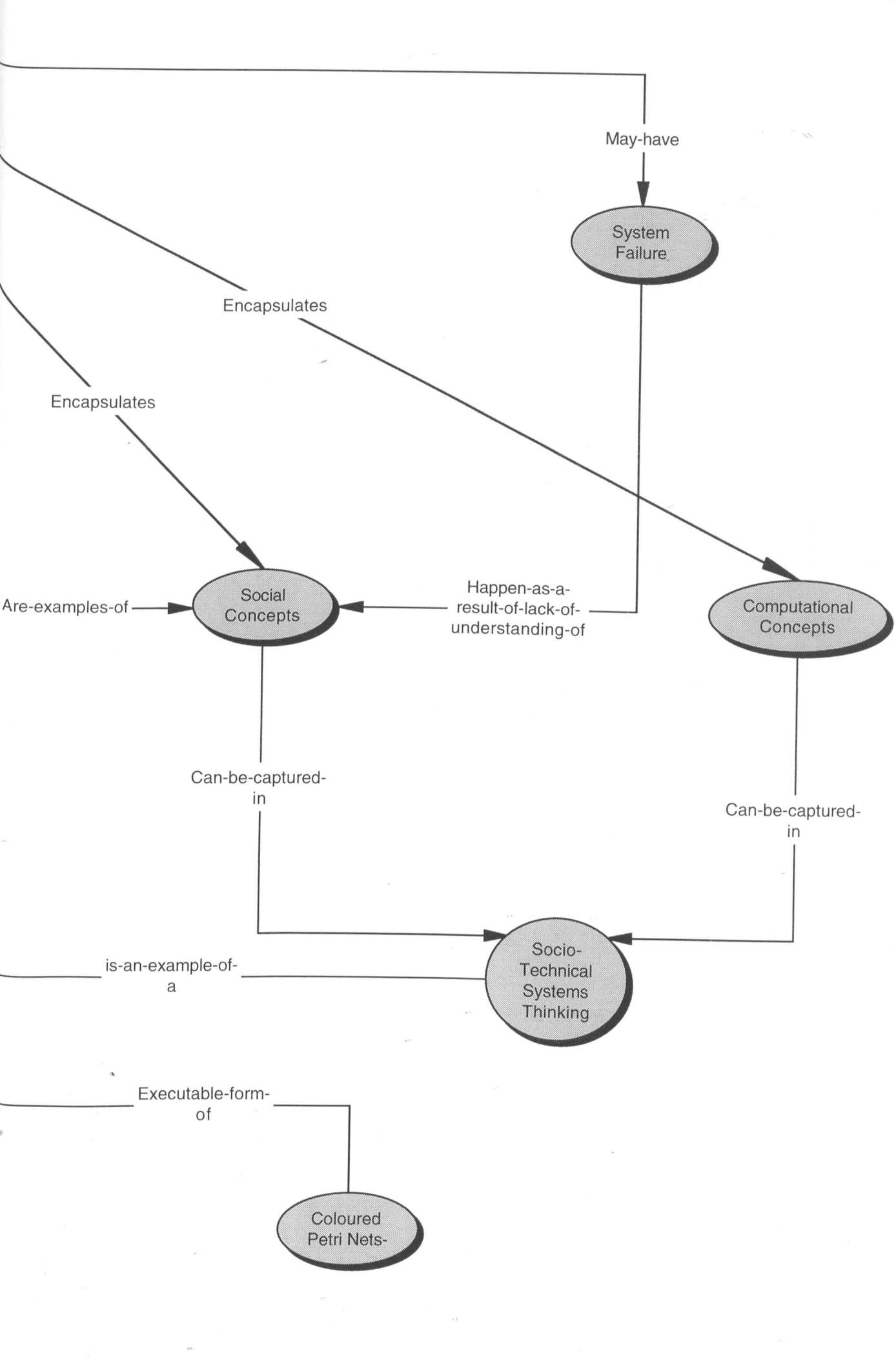
$$\forall a : \Delta_{Object} \mid a \; Utterance \; altered\_password$$
$$\bullet \; a \; Executor\,(system\_adm)$$

The above policy statement states that for every node in a conversation or interaction diagram which contains a utterance, the proposition of which is that the password file has been altered, the only actor which can execute that node is the system administrator.

Appendix D
A Semantic Net of the Thesis


A Set of Issues and Concerns
Defines
A General System Projection
Makes-Use-of
Is-an-instance-of
An Architecture
Consists-of
Consists-of
Semiotic Component
Is-part-of
Hermeneutical Component
Consists-of
Conceptual Framework
Is-part-of
Are-encapsulated-in
Is-comprised-of
Contains
Is-comprised-of
Syntactic Concepts
Contains
Are-embodied-in
A Modelling Language
Consist-of
Contains
Allow-for-the-analysis-of-the-strcuture-of
Modelling Method
Contains
Process Model
Supports-the-use-of
Contains
Is-used-in-the-construction-of
Makes-use-of
Guidelines, Rules, Recipes and Heuristics
Prefabrications
Supports-the-creation-&-use-of
Consist-of


An Information Technology System
Is-an-instance-of
Information Projection
Enterprise Projection
Is-an-attribute-of
Is-an-attribute-of
Mediates
Proposition
Responsibility
Is-an-instance-of-an
In-an-instance-of-an
Social relationships and values
Are
Are-encapsulated-in
Axiological Concepts
Allow-for-the-analysis-of-the-flow-of
Are-embodied-in
Is-used-in-the-construction-of
Are-encapsulated-in
Allow-for-the-analysis-of-the-strcuture-of
Allow-for-the-analysis-of-the-flow-of
A Static Model
A Dynamic Model
Is-used-in-the-construction-of
Is-an-adaptations-and-instantiations-of
Is-an-adaptations-and-instantiations-of


May-have
System Failure
Encapsulates
Encapsulates
Social Concepts
Are-examples-of
Happen-as-a-result-of-lack-of-understanding-of
Computational Concepts
Can-be-captured-in
Can-be-captured-in
Socio-Technical Systems Thinking
is-an-example-of-a
Executable-form-of
Coloured Petri Nets-