Supporting Design Understanding in Evolutionary Prototyping

An application of change theory and semiotics

Amir Albadvi
Information Systems Department
London School of Economics and Political Science

Submitted for the Degree of Doctor of Philosophy
University of London
1997

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Abstract

This thesis researches the problem of building design understanding in rapidly changing environments. Although evolutionary prototyping has been proposed before as a solution, little serious investigation has been undertaken into its practical and theoretical adequacy. This thesis assesses the evolutionary development approach and on the basis of the findings of an exploratory case study conducted in a large car manufacturer company, proposes a new perspective in this approach. It combines the planned organisational change theory and semiotics which respectively underpin implementation management and design understanding.

The cornerstone of the proposed perspective is a semantic analysis technique which complements evolutionary prototyping. The perspective builds on three cycles of planned change model: a vision cycle providing easy access to design knowledge, an action cycle supporting modular development of prototypes based on the semantics of design knowledge, and a fusion cycle institutionalising design understanding. An explanatory empirical study conducted in a management consultancy, provides a first step towards a subjective validation of the proposed approach.

A conceptual training process is suggested as a means of partnership between designer and user. This process provides a way for both user and designer to find a common designation for the terms they share in their communication, and to build a shared meaning and interpretation of actions in the workplace.

Keywords: evolutionary development, prototyping, semiotics, semantic agent-based modelling technique, planned change theory, design explanation, design understanding, conceptual training, user participation.

Acknowledgements:

I would firstly like to express my thanks and gratitude to my supervisor, Dr James Backhouse, who has never ceased to amaze me with his astute comments and suggestions.

For constant support and encouragement and from the deepest part of my heart, I wish to thank my best friend and my wife, Shahla, whose help I could not have done without. Only due to her undying patience, understanding and devotion have I been successful. She has sacrificed much and I dedicate this dissertation to her. There are two special young people to thank as well, Elham and Parham, who often had to do with less attention from their father than any of us would have preferred.

I am very grateful to Dr Jonathan Liebenau and Dr Edgar Whitley for their careful reading of the draft manuscript and for many lively discussions that have often helped to crystallise ideas and made my stay at LSE that much more enjoyable.

Declaration:

This thesis is entirely the result of my work and includes nothing which is the result of work done in collaboration. Any reference to the work of other researchers is clearly indicated in the text. This thesis has not been submitted in whole or in part as consideration for any other degree or qualification at the University of London or any other Institute of Learning.

Table of Contents

Chapter 1: Introduction and research approach

Chapter 2: The evolutionary development process

Chapter 3: A theoretical framework for evolutionary development

Chapter 4: Exploratory empirical study

Chapter 5: A method for evolutionary development

Chapter 6: Explanatory empirical study

List of figures

List of tables

CHAPTER 1

Introduction and research approach

Chapter overview

This thesis explores the potential for a new perspective on evolutionary information systems development. By providing semantic knowledge in an accessible form, the new perspective aims to facilitate the evolution of complex and uncertain systems in business change environment. This is the opening chapter of this thesis. The motivation and objectives of the research are discussed in sections 1.2 and 1.3 respectively. Changes in the business environment and the potential for an evolutionary prototyping approach to deal explicitly with these changes are the main impetus behind this research. The motive is to enhance design understanding in the development of dynamic information systems. The main objective of this work is to investigate the way in which a more supportive evolutionary environment can be developed so that designers may be capable of dealing with the changes in information requirements. Section 1.4 defines the basic terms used throughout this thesis. Section 1.5 sets out the rationale behind the research method we have employed. This section reviews the subjective/argumentative research method in building new approaches and techniques for information systems implementation. Three stages of this research are presented and also illustrated in figure 1.1 defining description, development, and justification in order to distinguish between theory building and theory testing phases of the research. Short-circuiting any one of these stages can result in dysfunctional research activities which produce war stories, black boxes, or ivory-tower prescriptions (Meredith, 1993). A convincing descriptive and logical argument was made about subjective justification for the research, based on the beliefs and assumptions about the better way of developing information systems. Subsection 1.5.4 argues that this research still needs further important work by case researchers, action researchers, and critical theorists to allow field experience and field data to generate greater refinement of the research result. This subsection also suggests a methodological long-term perspective of such an effort in a continuation of this research. Last, but not least, section 1.6 outlines the content of this thesis using figure 1.2 and its main contributions. It submits the major contribution of the research as the application of planned change theory and semiotic theory in evolutionary information systems development.

1.1 Introduction

Rapid change in the business environment means that organisations now face entirely new situations for which operating rules and procedures have to be supplemented by integrative devices, hence increasing the need for coordination. The growing diversity of the parts of an organisation increases the amount of information required for coordination and integration. Indeed, it is information that forms a vital component for the survival and continued development of any organisation. In view of the increasing importance of information, information technology is believed to be instrumental in transforming organisations (Child, 1988; Drucker, 1988).

Organisations are finding that they are now considering and using information as a resource, as a means of production, and as a strategic component. The increasing use of information technology to tap this resource has brought about the need for new implementation methods. When developing information systems, the question is how to implement appropriate systems within the context of the prevailing change in the business environment. It thus became apparent that in order to implement usable information systems, they have to be consistent with the organisational context within which they are intended to function. Much work has been done to this end. Much research and practice has been carried out; and a plethora of methods, tools and techniques have been produced. These range from hard-core structured systems analysis and design methods emphasising the technological content within which the information systems are being implemented, to soft information systems analysis and design methods that focus on the organisation or human context, and place considerable emphasis on getting the requirements right.

The prototyping approach offers responses to some of those concerns. With the increased involvement of end users in the development of information systems, the popularity of prototyping methods made it easier for systems to be built and refined iteratively. A number of methods and supporting tools and techniques fall into the general category of the prototyping approach. This research is a contribution to this category. It aims to marry subjective organisational views to objective technical issues of the prototyping approach. It proposes a method for a balanced design in which the

technological constraints as well as the human and organisational restrictions are considered.

1.2 Motivation

Requirements understanding has for a long time been recognised as an important attribute for good information systems design. This is not surprising since design understanding has been loosely equated with a designer's ability to ask sensible questions of users, to make good decisions based on incomplete data and knowledge, to produce practical designs, to diagnose the causes of system failures, and to be able to justify design decisions to users. The problem that information systems design methods and techniques are not supporting thoroughly the designers' understanding of the system under study is therefore of great concern. However, with advances in information technology, there has been great progress in computing techniques and the software solutions available. These offer excellent opportunities for developing more supportive systems. A fundamental question exists, however, as to what form this support should take.

To date there seems to have been a tendency to take each new offering that computer scientists provide, such as logic programming, object-oriented systems, functional languages and expert systems, and apply it to problems in information systems design, and prototyping techniques have been no exception. Such techniques have been applied largely without serious consideration of the fundamental problems involved in information systems design tasks and hence of the best way in which these techniques could be used. The prevailing changes in the business environment have also evoked great concern over the importance of prototyping techniques in general, and of evolutionary prototyping approaches in particular, as one of the most appropriate approaches for designing systems in this environment.

During the past decade the complexity of design tasks has led to development of support for the procedures that constitute the tasks themselves. In contrast to hard-core technical approaches to information systems design, some commentators (Avison &

Wood-Harper, 1990; Checkland, 1981; Checkland & Scholes, 1990) advanced the provision of other forms of soft support for understanding requirements. These two approaches run parallel with Habermas'(1972) theory of knowledge interests, namely with knowledge that helps to achieve human understanding, and with knowledge that helps to achieve technical control. The managerial use of the information systems is driven by the need to understand the meaning of data for action, whereas the technical operation of the information systems is driven by the need to control the data and maintain its consistency to allow correct program behaviour (Lyytinen, 1987).

Business change and its impact on changing information requirements require that information systems development is not just a matter of (software) technology change. Information systems development brings in its train both social and cultural change. Therefore, the choice of information systems design is not the question of one or the other but a question of the balance between them.

Therefore, there is a need to investigate the practical adequacy and theoretical utility of evolutionary information systems design methods. The focus of this analysis is on the evolutionary development approach using the prototyping technique and on how the approach can be supported by a framework into which social concerns of business change can be also fitted, so that integrated systems can be understood, developed and demonstrated. It is highly desirable that this analysis and design framework should provide more supportive environment so that it has the potential for helping the user and analyst both to increase their understanding of the system under study.

1.3 Objective of the research

The major objective of this work is to investigate the way in which evolutionary information systems design can be supported in a changing business environment. The difficulty with rapid changes in business is that while introducing an information system induces a change in some current state of the organisation, the evaluation and expectations of the organisation have themselves in the meantime changed. The amount of change affecting organisations today means that there is no longer a state of

equilibrium and we have to learn to design systems which cope with instability. This problem continues to stymie the development of successful information systems. Evolutionary prototyping offers a technical approach for improving this difficulty.

To achieve the main objective of developing a more supportive evolutionary environment requires an in depth study of the evolutionary prototyping approach, in particular, of the constraints encountered when implementing this approach, of the problem designers face in eliciting changing requirements using prototyping and of the design understanding the approach brings to bear. In this thesis, a study of these important issues is undertaken.

In the adopted research method a critical sub-goal is to determine important features of design problems in business environment and of methods needed to overcome them. Any partnership of designer and user within the evolutionary prototyping framework must be capable of dealing with these problems and may, therefore, incorporate design understanding techniques. Another crucial sub-goal is to study a supportive modelling technique that encapsulates design understanding. Using this technique the roles that designer and user should take in an effective partnership can be planned. Little work has been reported on planning the evolutionary process and managing it in changing conditions. It is essential that this be achieved, however, if the support environment for evolutionary development that can conduct change planning is to be constructed. Crucial to the present research is the provision of a planned change approach for developing evolutionary information systems.

The development of a more supportive design environment involves recognising and providing conceptual training for the system under study. A new concept of partnership between designer and user can be described by offering a conceptual training process. This must be based on a sound representation of the users' knowledge of the workplace, which in turn requires a sound theoretical base for developing a modelling technique. The study of the basic nature of the user's knowledge and how it can be represented is, therefore, an important objective of this work.

Knowledge transfer between agents[1] in the workplace (we will argue that this applies to the development of understanding) depends on some form of language construct. A critical objective of this work is the study of a set of scripts for conceptual modelling which can be used to identify the constructs necessary for expressing user's knowledge. These scripts should be able to represent the socially-constructed user's language to allow the knowledge to be easily expressed by users, captured in a schema, accessed by other users and hence transferred. It will be argued that satisfying this objective will satisfy the goal of providing support for design understanding and conceptual training.

1.4 Basic Terminology

This section will elaborate a set of basic terms about information systems development which we use throughout the thesis. We describe the key features of the business change environment and information systems design by postulating that information systems development is a form of object system change. This helps to focus on the features of object systems, their content and representation as forms, and to define the concept of a systems development methodology as a social institution (knowledge plus resources) which conditions and guides the perception, analysis, synthesis, evaluation and implementation of object system changes.

1.4.1 Business change environment

Today we are witnessing a transition from a predictable business environment to one where the greatest certainty is change itself (Backhouse & Albadvi, 1994). To be successful in a global competitive environment, organisations have to compete and win across the board. Not only do they need simultaneously to deliver high quality and low cost, which were once regarded as trade-offs, but they also need to be innovative, flexible and fast to the market on a continuing basis. The bundling of internal resources called for by this prevailing environment of heightened competitiveness is potentially very different from the configurations that allowed organisations to be successful in previous eras (Burn, Galliers, & Sauer, 1995).

[1] The term agent is used in this work to refer to one who (or that) commits intentional acts.

Today's economic situation is characterised by increasingly competitiveness, due to world-wide, fast-reaction competition and process/product innovation competition. The present business environment embodies dynamic and continuous changes. In order to address new competitive requirements, companies are realizing new important changes in their internal organisation, as well as in inter-firms links. Environmental uncertainty and dynamism have grown strongly, so that business researchers and managers use the word "turbulence" to refer to the new emerging environment. We refer to this environment as *business change environment* throughout this thesis.

The analysis of new inter-firm interactions[2] and of innovations in internal organisation[3] reveals a need for a continuous exchange of ever greater amounts of information. According to this analysis, information technology is fundamental in supporting and in fostering new inter-firm interactions and emerging organisational innovations (Ferioli & Migiarese, 1995).

The business change environment demands the exchange of significant volumes of information which in turn increases the problems of complexity and uncertainty that companies have to face. These problems, which will be discussed in chapter 2, stymie the requirements determination process in information systems development.

1.4.2 Information systems development

We define information systems development as (Welke 1983): *a change process* taken with respect to object systems in a set of environments by a development group to achieve or maintain some objectives.

Object systems consist of phenomena perceived by members of the development group. What is perceived is socially constructed through sense-making and institutionalised conventions. The concept of sense-making is defined as the mode in which a group interacts to interpret their environment and arrive at socially shared meanings. Object

---

[2] see Williamson model (Williamson, 1979) for inter-firm interactions

[3] see Galbraith information-based model (Galbraith, 1973; Galbraith, 1977) for a detailed analysis

systems identify a target of change. In general, there is more than one object system which a development group can identify. Object systems are often related, so that a change in one can induce a change in others. Designer and user perceptions of object systems need not coincide. This raises the issue of how to handle ambiguous or conflicting views of object systems throughout systems development. Object systems can be further characterised in terms of their underlying concept structure, representation form and ontology. The approach of this thesis to these issues is discussed further in chapter 3.

Information systems development is intentional, to the extent it reflects a planned change. It is based on developers' intentions to change object systems towards desired ends. Intersubjectivity means that the change process is founded on recognition of phenomena by more than one participant and on mutual understandings and coordination of participants' actions. Systems development cannot be just an artificial intervention because it always has to be embedded in a social and cultural milieu entailing many uncertainties. Therefore, the change process is not a deterministic one. For example, developers are often uncertain whether the planned intervention can be carried out, and whether the resulting object systems will have the desired properties (Hirschheim, Klein, & Lyytinen, 1995). The proposed framework of this thesis for a planned change approach in information systems development is discussed in chapter 3.

1.4.3 Conceptual model and schema

Systems developers must find an explicit representation for object systems, after they have been identified, to communicate them to others and themselves in the development group. Prototyping is one form of object system representation. Object systems can be represented in multiple ways. The chosen representation form depends primarily on the concept structure and its degree of accuracy and formality. Different requirements representation techniques are discussed in the next chapter.

Applying representation forms results in object system representations which correspond to information systems models as used in some parts of the information systems and

software engineering literature. *Conceptual modelling* produces changes in the representational forms, structure, and use of language (language change) that form the environment for communications through information systems use. Throughout this thesis, we use the terms conceptual model or information model[4] in order to refer to a set of conceptual and notational conventions which help to perceive, organise and specify some data.

In the literature, the term data modelling is also used with the same meaning. Data modelling, however, deals not only with linguistic issues but also technical ones such as data structures and storage organisation. In the sense of conceptual modelling, the term data model is sometimes combined with the word language, as for example in data modelling language or data description language (DDL). When a conceptual model refers to the outcome of using a modelling language in some specific situation, we use the term *schema.* To clarify the difference between the two meanings, consider the following example. In accounting, the conceptual model (data modelling language) consists of the terms and principles of double-entry bookkeeping that guide our perception and arrangements of economic data. The schema is the chart of accounts for a specific company (Hirschheim, Klein, & Lyytinen, 1995). The distinction between conceptual model and its use (in a schema) implies that the first must exist before the second can be created. Usually they are also developed by different people. Conceptual modelling is the activity of creating a *conceptual model* (in the sense of a schema). If the model becomes accepted by the organisation it will produce changes in the organisational knowledge base. Hence, we shall consider conceptual modelling as a change process.

1.4.4 User, analyst and responsible agents

The term user is often a catch-all for anyone who works with the system who is not part of the technical team and unlikely to be an expert in computing (Avison & Fitzgerald, 1995). In any information system development project, it is crucial to

[4] We also use the term 'enterprise information model' to refer to a class of conceptual models which covers the whole organisational activities. Again this term connotes that part of data modelling which deals only with linguistic modelling.

identify the potential users of the object system. Different information systems development approaches have different assumptions and definitions for user, analyst or system developer.

Throughout this thesis, we use the term user as organisational agents who interpret and make sense of their surroundings. The analyst is the *change agent* who helps the users make sense of the new system and its environment. This assumption presupposes a working environment where the users and analysts work as a team rather than as expert and non-expert. It rejects the idea of representative participation (Mumford, 1983) by management assuming that those representatives do indeed represent the interests of all users affected by design decisions. Managers are not necessarily the only organisational members who know systems objectives. The socially constructed view of systems objectives taken in this thesis suggests that there is no single reality which can be represented by one group of organisational members. Management, too, tries to make sense of the confusion with the commitment to the organisational missions. Systems objectives are not given, but constantly evolving.

The role of the analyst is to interact with all potential users to find out what type of system makes sense, but there is no objective criterion which distinguishes good from bad systems. It all depends on what the parties come to believe to be appropriate. The analyst should work from within the users' perspective and help them to find their preferred view. He should ease the transition from one view point to another, thereby alleviating possible resistance to change. Ideally, analysts or systems developers are able to reduce the pains of change. In this change process, we emphasize on the process of negotiation of meanings and clarification of responsibilities.

The approach of this thesis is consensus participation (Mumford, 1983) of all users throughout the design process. It has the merit of making the design decisions by all users on the basis of an agreed responsibility structure. An immediate consequence of this assumption is that active *responsible agents* are always included into the syntax and semantics of any representation of the system under study. These responsible agents are those users who construct the social world and hold responsibilities for their actions.

To achieve consensus about responsibility structure, continuous interaction among all parties is crucial.

Consensus participation among all the users, with the help of the analyst as facilitator, provides the process of increasing mutual understanding. One criticism to this view is that it is difficult to know how close or far these understandings are from each other. Some people would claim that the goal of a single unified interpretation of the responsibility structure is illusory and no formal representation can resolve the problem and challenge of different understandings. This argument concerns the role of power in organisations. Although we believe that power is always a factor in a change situation (Mumford, 1996) and a principal obstacle to genuine participation, we do not intend to study this issue in this research. Our assumption is that with a balance of power within the organisational structure, the semantic gap problems are supposed to be resolved through rational communication, given sufficient time. But where there is power asymmetry, there is greater value in being able to reach agreement in a systematic and non-disruptive manner.

1.5 Research approach

The practice of research is a messy and untidy business which rarely conforms to the models set down in methodology textbooks (Brannen, 1992). This research can be broadly classified as a **qualitative research** which through a wider lens is searching for patterns of inter-relationship between a previously unspecified set of concepts.

The research was begun by defining general concepts on the shortcomings of evolutionary prototyping approach in information systems development methods which, as the research progressed, changed in their definitions. Very soon it became clear that the modes for traditional empirical approaches, mostly based on observation, do not represent well the research agenda with an integrative view of information systems development methods. The object of the research is to evaluate an information systems development method, hence more interpretations for understanding observations will be needed.

This research adapts to the newer research modes based on the proposed taxonomy of information systems research approaches by Galliers and Land (1987). Using that taxonomy, the present research is classified in the **subjective/argumentative** category which was first defined by Vogel and Wetherbe (1984) as "capturing creative MIS research based more on opinion and speculation than observation". In their view, this category complements empirical research methods by extending the domain of MIS research beyond that based on observation. The research method designed for this research emphasizes socio-technical interpretation over the entire information systems development process. While the basis of concept formulation is an argumentative description, empirical case studies and basic research on related literatures are also used as supports for the main argument. Three stages in the research cycle of this particular research have been identified: description, development and validation. The three stages as three dimensions of the research effort used for understanding the research method applied in this study can be presented in a time sequence order. Figure 1.1 shows the suggested relationships.

1.5.1 Stage one: description

This stage is *inductive* and is begun with an argumentative description about the main shortcomings of the evolutionary prototyping approach in software systems development. The description stage of the research sets out to assess the evolutionary prototyping approach to information systems development for its practical adequacy and theoretical utility. Although the evolutionary approach gave the software industry the technical capacity to develop more relevant evolving systems, the assumptions behind the approach are found to be very implementation-oriented. This problem continues to stymie the development of successful information systems. This difficulty is rooted in the use of a physical scientific model as a seminal theory for evolutionary software development methods.

At the beginning of this stage, the basic research of the literature has been conducted covering four major topics: information systems development methods with particular emphasis on prototyping approach, semiotic theory and semantic analysis technique, organisational change theories and theories on the possible levels of change, and finally

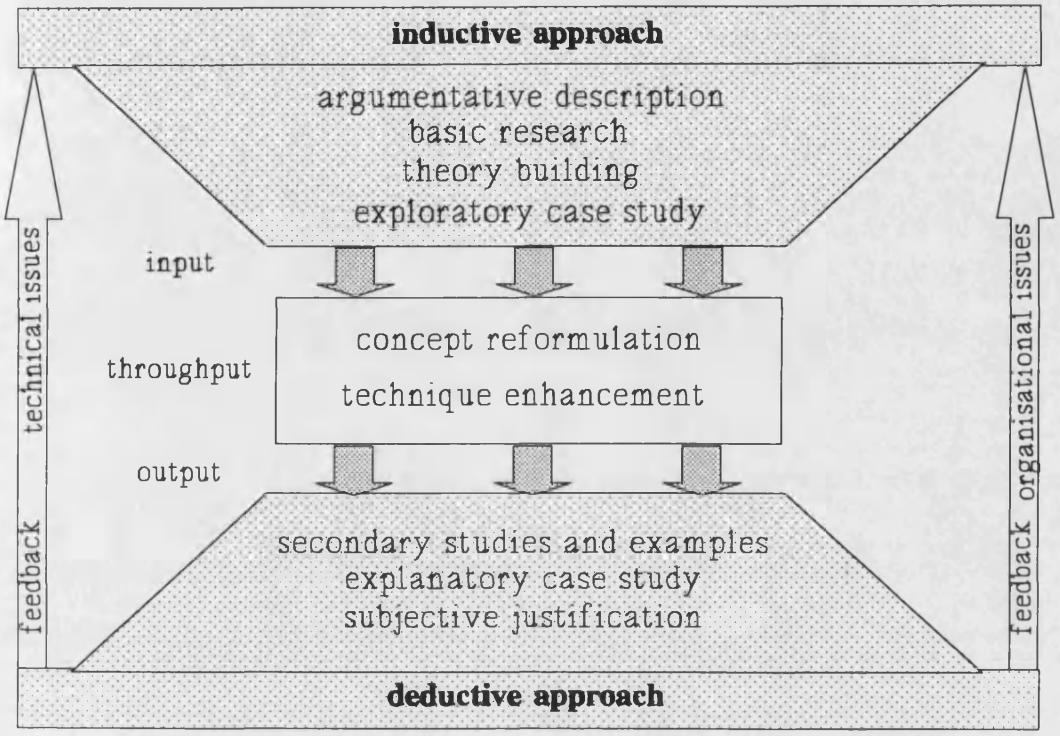


Figure 1.1 An illustrative model for understanding the research method

process theories.

To complete this stage and link it to the development stage of the research, an exploratory case study in evolutionary prototyping approach for the development of a highly complicated software system in a car manufacturer company was also conducted to analyze the argument in the real world. This case study, employing participant observation of the development environment and in-depth interviews with developers, had an exploratory purpose to gain a better understanding of the key issues related to the success of evolutionary prototyping. The analytical induction, which is simply descriptive, takes the research focus from the findings of the case study through the formulation of concepts to their validation and verification. Findings of the exploratory case study pinpointed two *lacunae* in the evolutionary development: the lack of an

effective implementation process management, and the use of hard systems thinking without having a support model as a frame of reference for analysis and design.

1.5.2 Stage two: development

While at the start, the research problem was only roughly defined, the exploratory case study from the description stage inspected those features which were essential to the problem definition and through an induction process generalized them by abstraction (Denzin, 1970; Znaniecki, 1934). Injecting theory into the findings of the case study from the results of the literature survey provided a working outline for the research problem, which was then formulated. A new perspective on an evolutionary prototyping approach, using theories of change for planning the whole process of implementation and using semantic analysis as an explanatory technique, form the expected logic of inquiry in the development stage.

The proposed perspective to evolutionary development is supported by a theoretical framework. The framework consists of two theories: planned organisational change theory and semiotic theory. Both theories adapt to the key building blocks of evolutionary approach, e.g. cooperative design, facilities for learning and communication, improving user understanding. Although both theories assume socio-technical approach to information systems development, they employ different assumptions in dealing with information systems as a social design problem. Planned organisational change model takes a process-oriented view to information systems development. It emphasizes mainly how to go about developing an information system and concentrates on means to achieve systems objectives but not ends. The assumptions behind the theory is that the system objectives are legitimate and agreed. The main weakness of this theory is the failure to focus on the legitimation of the ends in information systems project.

In contrast, semiotic theory by employing the semantic analysis technique offers an analytical tool to understand a complicated web of organisational behaviours which link different concepts and actions into a rich array of varying phenomena in a socially constructed world. It continuously strives for consensus where there are found to be

ambiguities and conflicts in requirements and helps to achieve more stable, extendable and flexible semantic schemata of the language used in the workplace. The technique explicitly analyzes the multiple linguistic barriers to learning and communication that may exist both within and beyond the immediate work situation. Of particular concern is the recognition that work practices are connected to different work languages and to change either one means a change in "forms of life".

The above descriptions suggest that planned organisational change theory and semiotic theory can complement each other and form one theoretical framework in response to the main shortcomings of evolutionary approach. Planned organisational model provides the means necessary to manage the implementation process, while semantic analysis technique aims at discussing ends and creating a common understanding among all users. Semantic analysis implicitly assumes that ends in any information systems development are conflictual, ambiguous and a subject of considerable disagreement and debate. Requirements are seen as emerging from interaction between user and analyst as both try to understand the organisational situation and make sense of it. The management of this emerging process is the concern of the planned change model.

The idea behind the proposed theoretical framework implies some type of coherence and integration between two theories. Both theories share some concepts and beliefs on the change process and also complement each other on achieving a specific change in support of an information system development. Therefore they offer more while together than separate. The proposed theoretical framework supports a new method as a well-defined description of the techniques employed in the proposed perspective to evolutionary development. The method addresses itself to the middle stages of information systems development covering analysis and design. By taking a socio-technical view, the new method suggests that information systems development should lead to both an optimal social as well as technical system.

1.5.3 Stage three: validation

The issue of the generalizability of findings in a broader context raises the question of the validation of the research results. During the early stage of research the problem

of assessing the value of the new approach to evolutionary development and of the generalizability of the research results to other development environments was foremost. At first sight, engaging with the issue of validation seemed very difficult and somehow impossible. The researcher felt like a person painting himself into a corner! But soon an investigation of similar efforts for developing new paradigms and techniques in software engineering showed how it might be done.

Fitzgerald (1991) addresses carefully this issue in his research for the development of a technique called "action modelling". He defines the term validation as "the justification of the technique (or approach) in terms of its power, effectiveness and practicality in relation to its purpose and objective". He examined the problem of justification for a new technique addressing different possible answers and angles. The reductionist approach of scientific method for breaking a problem down into smaller parts for examination and explanation was investigated and was found wanting. He concluded that it was difficult to conduct laboratory experiments using the proposed technique to model the real world, or even to model an artificial situation. We can model example situations using the proposed perspective in evolutionary development, but comparing the results with other approaches in order to justify the new approach is very difficult because we enter the realms of subjectivity. Fitzgerald (1991) looked at a number of papers from the discipline of computer engineering and discovered that, in general, there is no evidence of seeking validation for new techniques and approaches in software specification techniques in any more satisfactory way. In his words, "it was really a surprise that so many argued that their techniques were useful without really undertaking studies to prove it in any way." The examination of papers on new software engineering techniques shows an almost total absence of any attempt for validation of this type of research. He also mentioned that, in spite of the assertions by practitioners that the major way to validate is by constant use and reuse, this merely determines what is popular but it certainly does not determine what is best. Jeffrey (1987) argues that the excuse of the immaturity of information systems studies as a youthful discipline avoids us conducting the testing and validation stage for new techniques and approaches. It perhaps can only be undertaken after maturity, i.e. when hypotheses in this discipline have been proven!

Finally, Fitzgerald (1991) asks us to recognise that the methods of validation which can be selected will be **subjective justification** for the product of the research and that the best that can be achieved may well be a circular validation based on the researcher's own beliefs and assumptions. Turner(1967) has called it "contextual justification" with great emphasis upon validation within the context and assumptions. He stated that we need not, and should not, insist on any ultimate justification which does not exist.

We need to be more rigorous in the area of subjective justification and at this stage of research a *method of triangulation* can be adopted implying the use of multiple research strategies (Burgess, 1982; Burgess, 1984) to tackle the problem of validating new perspectives in information systems development. Method triangulation can be between-methods using different methods in relation to the same object of study. Denzin (1970), in his original formulation of triangulation, saw the combining of research strategies as a means of examining the same research problem and hence enhancing claims concerning the validity of the conclusions that could be reached (Brannen, 1992). It is also mentioned in Rieger & Wong-Rieger (1988) that the different methodological approaches tended to compensate for each other's deficiencies and lent confidence to the validity of the findings. Fitzgerald (1991) refers to this as *a pluralistic approach*, using a number of different approaches to help validation. The argument is that researchers ought to select a range of methods that are appropriate to the problem of justification of the research. Therefore, this stage of research represents a fundamentally different type of inquiry compared to the description stage. Here we need to employ a *deductive* approach, by starting with a well-articulated general model resulting from the development stage, and then by collecting data to test the propositions put forward in that model, using different approaches and methods.

The findings of the exploratory case study guided the research focus to the change process theories and their application in the management of information systems development. A number of researchers have studied planned organisational change theory and its effectiveness in managing an information systems development project. Therefore, a review of those research results as secondary materials have been undertaken in the validation stage of this research. The aim is to study the main

requirements of the planned change model and how they coordinate with characteristics of semantic analysis technique.

An explanatory case study was also conducted to demonstrate the power of semantic analysis in role clarification and conceptual training of people, especially when organisational change is going to happen. This case study, which explains the semantics of the substantive business of one of the biggest management consultancy groups, showed the roots of conflicts in meaning and application of terms within the company. This case study directly addresses the usefulness and practicality of semantic analysis in role clarification and responsibility negotiation, and also in providing a supportive model for conceptual training specifically at the early stage of implementation of any organisational change. The findings of the case study shows that the characteristics of the semantic analysis technique addresses the need ,found by other researchers, for a successful planned change approach. Indeed this case study had an explanatory function about the benefits of semantic analysis, through viewing the in-depth character of change in the organisations and also by mapping the norm structure in relation to the rule-guided activities, with norms acting as the meta-rules of the system. The case study was deliberately selected without any implication of developing a software system, in order to examine the independence of semantic model of any functional or procedural detail, and the ability of this technique to specify the underlying prime business tasks and information needs of the organisation, without committing to any particular computer model. This case study points to the power, effectiveness and practicality of this technique in relation to its purpose as an explanatory prototyping tool.

The use of an example to show how semantic analysis technique can be applied will be used as a way to explain better the technique, with the understanding that this simply helps to illustrate the technique rather than to validate it. Strictly speaking, by using examples it cannot really be said that the proposed model for information systems development has been proved, even if it can repeatedly be shown by examples to be effective and useful. What is intended is that the research hypothesis can be shown to be difficult to refute. For this purpose we adapted an example by Rumbaugh *et. al.*

(1991, p.151) from their book "Object-oriented modelling and design".

Finally, it is necessary to mention the importance of using empirical real world projects as another strategy to provide better justification of the proposed perspective. Unfortunately, owing to its inherent difficulties such as uncontrollabity in a specific period of time, unpredictability and extensive resource consuming, this strategy could not be easily fitted in the limited time frame of this research. However, according to Scott Morton's (1984) research typology, real world projects should eventually be used to evaluate the effectiveness of any new concept, but ought to be left for further research.

1.5.4 Method of further research

Some methodologists or researchers approach research from an either-or framework. They make rigid distinctions between basic research and applied research. This framework dislocates the close connection between situated practice and the generation of theory (Suchan, 1993). Hence it is necessary to present an outline of the future agenda for further efforts in continuation of this research from the viewpoints of both applied and basic research concerns.

Given that any definitive validation for the research is impossible, any further research should be planned on the basis for designing and implementing a "validation endeavour". This is indeed possible through the adoption of a long-term perspective:

- developing a clear vision of the nature of the information systems development contexts and processes

- providing a complete and deep understanding of the inherent nature of the proposed perspective on evolutionary information system development

- assessing applications of the proposed approach in practice through field-based methodologies

- synthesizing and contrasting the results of these assessments, given the growing body of literature regarding development contexts and processes

- discovering and expressing the strengths and weaknesses of the proposed approach

- enhancing and revising the body of literature on information systems development contexts and processes

Through the above process of steady accumulation of evidence and paradigms regarding information systems development methods, we can indeed lay down a further research strategy to approach the validation of the research. It is practical ...though slow and might be often tortuous.

1.6 Outline of the thesis and its contribution

In chapter 2 the study of evolutionary prototyping approach starts with a reassessment of problems of requirements determination. It argues that requirements determination problems can be described abstractly in terms of complexity and uncertainty. These generic problems are defined as: complexity- having too much knowledge/information and uncertainty- having too little knowledge/information. The ever growing competitiveness of the business environment means that new emerging interactions among organisational actors require a continuous exchange of ever more information. The turbulent conditions in the business change environment escalate the problems of uncertainty and complexity. A study of these dual problems and methods of overcoming them has led to the identification of prototyping as a more effective requirements representation technique compared with any text-based and interpretive model of requirements. In this chapter the requirements prototyping process will be discussed and then the evolutionary development approach as a completely independent approach to information systems development is described. A study of evolutionary development problems in a business change environment is undertaken. This study highlights the various aspects of problems in the approach and leads to the conclusion

that the problems can be described in terms of: (1) lack of process management system in evolutionary development approach, and (2) lack of a conceptual model for the systematic and rational selection of requirements to prototype, for an effective grouping of actual requirements into clusters and for an interactive traceability to original requirements statements.

The central theme of chapter 3 consists of a proposal for a new theoretical framework for evolutionary information systems design and development. The new theoretical framework is rooted in two theories: planned organisational change theory and semiotic theory, in response to the two shortcomings identified in the evolutionary development approach in chapters 2. These theories are presented in two parts of chapter 3. The first part of chapter 3 begins with the theoretical perspective of change and persistence. After defining different orders of change, we discuss process theories of change in relation to information systems development. Lewin's (1952) three-phased change process model - unfreezing, moving and refreezing - is suggested as a model for managing the implementation process in evolutionary development. The model uses prototyping in the moving cycle and applies a semantic analysis technique in the unfreezing and refreezing cycles. The proposed analysis technique is based on the application of semiotic theory in requirements elicitation. The second part of this chapter discusses semiotic theory and focuses on the users' own interpretations of what they do. This concept in requirements analysis can be applied by carefully studying the users' communication during work. The results of this can be used to design conceptual structures that fit into the language of users. From the study of language and how it relates to work situations using semiotic theory, requirements understanding emerges from discerning patterns of behaviour by organisational actors in their work situation. Chapter 3 also introduces the semantic agent-based modelling formalism and its graphical representation - the ontology chart - with the aid of a comprehensive example. This representation technique, it is argued, can provide design understanding and the potential for conceptual training. The semantic analysis technique appears to accord well with features of the planned change model proposed for managing the evolutionary development process. Chapter 3 lays down the theoretical foundation for a new perspective on evolutionary development proposed in this research.

Chapter 4 discusses a case study conducted in a large car manufacturer, which identifies constraints encountered when implementing evolutionary prototyping approach in a business change environment. This case study justifies the problem formulation investigated in the description and development stages of the research method. This exploratory case study investigates the identification of the main difficulties of using the evolutionary development approach in practice and the utility of the proposed theoretical framework to overcome those difficulties. The case studied is a highly complicated product definition system under development in one of the biggest car manufacturers in the U.K. The development method adopted is the evolutionary prototyping approach, using object-oriented techniques. The description of the case starts with the background of the car industry and the company in order to gain a better understanding of the magnitude of the problem and complexity of the environment. Then the characteristics of the business change environment and how it affects the development of a highly complicated information system are examined. This provides a clear idea of the business change environment in this industry and also the rationale behind the chosen case. The discussion leads on to the results and findings of the case study and puts them in the form of an argumentative description about the shortcomings of the evolutionary approach. Two critiques are developed of the principles of evolutionary development which underpin this conceptual practice. The chapter concludes that the lack of an effective implementation process management system and the lack of a support model for evolutionary development are tangible shortcomings in the evolutionary development approach, especially in regard to the changing business environments. On the basis of the findings of the case study, the main argument of the research is then formulated using the proposed theoretical framework in the previous chapter.

Chapter 5 offers the proposal of the research. It presents a new perspective on an evolutionary development approach based on the proposed theoretical framework. First an overview of the proposed approach is introduced in three main levels: organisational, conceptual and technical, showing the combination of top-down and bottom-up cycles in the proposed method. A detailed discussion follows of the new perspective to evolutionary development using semantic analysis and prototyping techniques within the

control of planned change model. Using the new perspective, a development method is introduced to formalise the stages and to merge them into a coherent whole. The new method consists of three cycles: vision, action and fusion corresponding to the three stages of the change process. The objective of the new method is not to offer step by step prescriptions in analysis and design, but is intended to cover just the most important issues raised in using the proposed perspective.

Chapter 6 discusses the issues related to the justification of the proposed method. First, it summarizes the results of the empirical studies carried out by other researchers as attempts to justify the application of the planned change model in managing information systems development. It then presents the findings from a second case study, conducted in one of the 'Big Six' management consultancies. It examines the relevance of a semantic agent-based modelling formalism for high level corporate modelling and its utility in vision and fusion cycles of the proposed method. It explains how the semantic analysis technique enforces reconciliation of data definitions across the organisation. The validation stage of the research method prescribes an explanatory case study in order to provide a better justification for the proposed perspective. The case studied is an important part of the company's corporate data model and explains the substantive business of one of the biggest management consultancy companies. The findings of the explanatory case study directly address the usefulness and practicality of semantic analysis in role clarification and responsibility negotiation, required in applying a planned change model. The findings are presented in the form of observations which show the benefits of the proposed method in response to the shortcomings of the evolutionary development found in the first case study presented in chapter 4.

Chapter 7 concludes this thesis and starts with the summary of the research. It then presents the outlook for further research.

The aim of each chapter in thesis can be mapped into the requirements of the research method. Figure 1.2 represents the contribution of each chapter in the designed research approach. The thesis can be read linearly from start and finish. Every effort has been made to ensure that this thesis perfectly reflects the research undertaken.

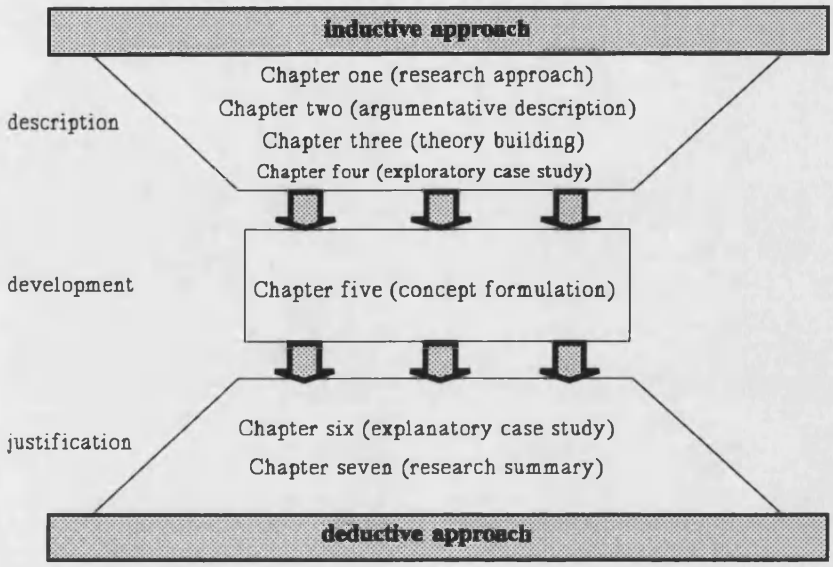


Figure 1.2 Mapping the outline of the thesis into the research method

The major contribution of this work is demonstrating the application of planned organisational change theory and semiotic theory in evolutionary information systems development. It is a new perspective on evolutionary development with a new kind of supportive design environment that results from applying these theories. This research also describes how such theories can complement each other in supporting evolutionary information systems development.

Another contribution of this research is the concept of clustering user requirements into subject areas based on their semantic properties. The current data modelling approaches offer subject-clustering, where entities are assigned to a specific subject group to allow easier navigation of the model. However a major problem lies in assigning entities arbitrarily to subjects, and in the inability to sustain consistently over time the categorisation of subjects. The proposed clustering concept in this research encounters no such difficulty, because the semantic constraints within the semantic agent-based modelling automatically associate each semantic element with its natural subject area. Therefore it provides us with a major complexity-reducing concept in modelling large-scale information systems.

Finally, a further contribution of this research is introducing to the evolutionary development the concept of an information system as a social institution. Whereas the evolutionary development approaches information systems as technical systems with social consequences, the new perspective of evolutionary development proposed in this research assumes information systems as social systems with technological implementations.

CHAPTER 2

The evolutionary development approach

Chapter Overview

The focus of this chapter is on the problems associated with evolutionary development. After a historical survey, section 2.3 discusses the dual problems of uncertainty and complexity in information requirements determination. The problems which will be intensified in a business change environment are those where a continuous exchange of ever more information is required. Complexity and uncertainty of systems can be reduced by requirements representation techniques. Section 2.4 discusses different requirements representation techniques among which requirements prototyping has the potential to cope with change in requirements. In section 2.5 requirements prototyping process and its origin in physical scientific approach to problem solving is presented. The subsequent section introduces the evolutionary development approach. The approach starts with gradual development of a requirements prototype and then allows the prototype to evolve continuously and be adapted in the use environment. This characteristic renders the approach viable as a strategy for system development in business change environment. Some evidence of the problems associated with evolutionary development in large system design is presented in section 2.7: difficulties with managing the implementation process, lack of a theoretical foundation for analysis of requirements statements, lack of a technique to assist in the discovery of requirements features and clustering subsections of a large system and finally lack of a technique to portray effectively the original requirements during prototype evaluation (traceability of requirements features). Section 2.8 outlines the main argument of this research. It states the difficulties in the evolutionary development approach for large scale systems in business change environment. The rest of the section 2.8 describes the overview of the research approach around two important issues: first, the development of a holistic process management system and second, the development of a support model for the systematic and rational selection of requirements to prototype. The support model also needs to sustain an effective means of grouping requirements into clusters and an interactive traceability to original requirements information. Section 2.9 concludes this chapter.

2.1 Introduction

As the reliance upon information has deepened and its value recognised, increasingly interest has focused on the development and integration of methods for constructing computerised information systems. The classic prescription for the development of information systems is to follow sequentially the stages of the system development life cycle. The information system analyst can choose among several methods that address individual stages of the systems development life cycle. Alternatively, a systems analyst can select one of the many integrated methods which address two or more of the stages.

One common shortcoming of most methods of the integrated system development life cycle is that they do not completely address the information requirements determination stage. The stage of determining enterprise information requirements is the most critical phase of the system development life cycle (Cooper & Swanson, 1979; Khan, 1985; Brooks, 1987). Therefore, a method which can simplify the information requirements determination phase would potentially result in enormous benefits for developers of computerised information systems as well as for users. Ironically, most integrated system development life cycle methods fail to capitalize on those potential benefits (Colter, 1982; Couger, 1982).

The classic life cycle approach follows sequentially a set of phases. Some alternatives to the approach have been developed (Ahituv & Neumann, 1990). The most popular is system prototyping for constructing computerised information systems (Naumman & Jenkins, 1982; Andrews, 1983; Appleton, 1983; Davis & Olson, 1985). In particular, developing an information system using evolutionary prototyping has rapidly gained acceptance as a preferred approach in business change environment (Connell & Shafer, 1995; Guimaraes, 1985; Young, 1984).

One reason for the popularity of requirements prototyping is that it allows for the information requirements determination to be handled iteratively as the system is constructed. This is done in response to the difficulty of determining a complete set of information requirements prior to starting the design stage of a system development

project. An information system analyst can construct a prototype of an enterprise information structure and be comfortable in the knowledge that the structure can be easily modified in subsequent iterations of the evolutionary process to incorporate any piece of information that was missed in earlier iterations (Kravshaar & Shirland, 1985; Naumman & Jenkins, 1982; Sroka & Rader, 1986; Wetherbe, 1982).

Conversely, one major disadvantage of evolutionary development approach is the difficulty of controlling it, especially for large projects (Alavi, 1984; Andrews, 1983; Dennis, Burns, & Gallupe, 1987; Mahmood, 1987; Pliskin & Shoval, 1987). In particular, it is difficult to determine which subsystem and how large a portion of information requirements needs to be addressed in each iteration of the prototype and how much need to be deferred to the next iteration. As a result, information systems scholars have been somewhat hesitant in recommending this approach as an alternative to the classic life cycle approach (Ahituv & Neumann, 1990).

This chapter focuses on the main challenges in evolutionary development approach and its difficulties in business change environment. It also provides an overview of a complementary approach to evolutionary development.

2.2 Historical survey

Insights into the historical development of information systems analysis and design methods can be gained by looking at the software development life cycle model. There are numerous models of the life cycle of system development. Some of these methods are descriptive (i.e., they describe what exists), some are prescriptive (i.e., they prescribe what steps should be taken), and some are normative (i.e. they establish a standard; for example, government regulations that set out specific development phases that must be followed).

As concern for the information systems development process has increased almost exponentially during past two decades, so has concern for having the right life cycle model. This concern had some very positive aspects and also some that may be

counterproductive. On the positive side, the backbone of any systematic, visible information systems development process has to be a clear set of workproduct definitions and an indication of what steps should be taken to create those workproducts; this is precisely what the life cycle model is intended to do and can do, if properly used.

The negative side of this concern over having the right life cycle was that much effort has been expended for little or no gain. It must be remembered that we are discussing models, not reality. Models, by definition, are an abstraction of reality. They can help us shape reality by helping us see the relationship between different aspects of development more clearly, and thus it is important that they be relatively accurate. However, since they are always simpler than reality (or should be!), we should not become too impatient when they seem not to capture everything we see in practice.

It seems that our collective understanding of the characteristics of information systems development life cycles is changing rapidly. In such a situation, we must be prepared to cope with models that always seem inadequate; eventually the rate of change of our understanding will slow down, and then we will be able more easily to prescribe appropriate life cycle models (Freeman, 1987).

2.2.1 Software development life cycle

The development of information intensive systems is comprised of a series of phases called the software development life cycle (SDLC), that provides a framework for the effective management, guidance, and control of the process. The advantages of this model is the breaking down of a large complex process into a manageable, well-defined series of small steps (Sage & Palmer, 1990). The classic representation or paradigm of the software development life cycle is the traditional waterfall model. The waterfall model has its roots in traditional systems engineering. Figure 2.1 presents the main stages of the waterfall model. It views the process as a set of phases or steps, starting at requirements analysis and capture, moving then to system design, on to implementation and testing and finally to system operation and maintenance. Although the model is viewed essentially as sequential, it should be noted that iteration will

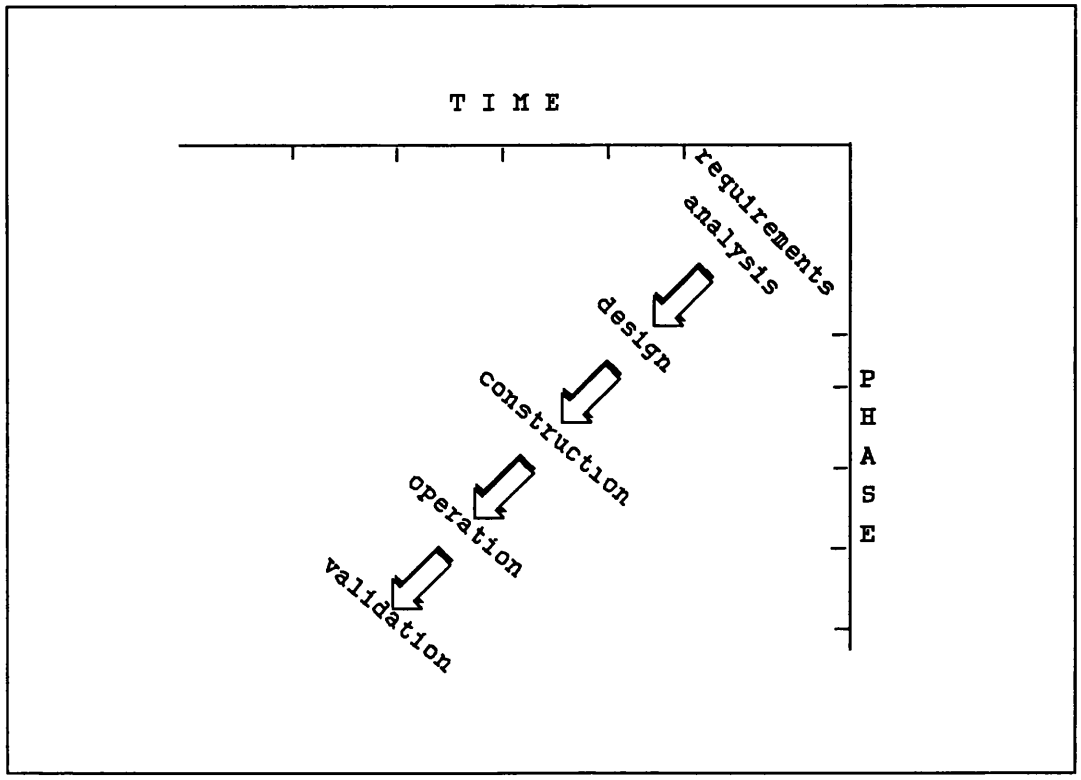


Figure 2.1 A waterfall model of the Software Development Life Cycle

almost certainly occur between the steps and within each step. In more detail these steps are:

• **System requirements analysis:** The first phase in any information systems development is the development of a set of information requirements, that will drive the remaining steps. In this step the overall system requirements and also the software requirements are detailed. The outcome of this step is a requirements specification document that is used to guide the rest of the development process. This thesis is specifically focused on this phase and will develop this phase in much greater detail later.

• **System design:** The translation of the requirements specifications into a design representation is the outcome of this step. Design is divided into a series of smaller steps which include the elaboration of a high level design model of the system and then the development of a detailed design model that maps directly to the implementation structures being developed.

Design traditionally focuses on the representation of data structure, software architecture and procedural details (Pressman, 1987).

• **Implementation and testing:** In this phase the design is translated into machine-readable form. This is generally accomplished by coding the detailed design in a high level or machine level programming language. The individual units of code are tested individually, integrated and tested as a group. The first form of testing is called unit testing, the second is referred to as integration testing.

• **Operation and maintenance:** The developed system is placed in operation and maintained. Maintenance is performed on the operational system in response to any errors that escaped the testing and review activity of the previous steps, as well as to implement changes so the system can remain responsive to changing conditions in its environment.

2.2.2 Requirements elicitation phase

The most difficult aspect of information systems development is eliciting the requirements for the system (Brooks, 1987). Requirements definition is performed in the earliest phase of the development life cycle and is commonly known as requirements elicitation (or requirements engineering). The requirements elicitation step is defined to include the capture, analysis, and definition of requirements or needs to be met by development of an information based system.

The requirements elicitation phase has come to be regarded the most important stage due to its overwhelming influence on system quality. Errors in requirements specification are the most costly, in financial terms, in terms of effort and time needed to correct, and in their impact on final user satisfaction (Boehm, 1984). Empirical evidence has constantly demonstrated that errors made in the requirements definition process exact a disproportional cost compared to errors made later in the life cycle. Studies (Boehm, 1976; Fagan, 1976) have shown that there are two orders of magnitude difference in the cost to repair an error made during requirements definition than in

repairing the same error near the end of system development. Historically, (Tavolato & Vincena, 1984) more than fifty percent of all detected errors are made during the requirements definition process. The impact of requirements errors result in creation of systems that will not satisfactorily address system needs (Davis, 1990b).

Table 2.1 indicates a set of characteristics that we look for in requirements information so we can produce a system that is both correct and what the user wants. These characteristics have been found to be of critical importance in determining whether a requirements specification represents an accurate representation of system needs (Davis, 1990a; Palmer & Fields, 1992; The Institute of Electrical and Electronic Engineers Inc., 1984). A set of system requirements specifications which can boast all these characteristics reduces the possibility of errors within the requirements and therefore the risk that the requirements will be inaccurately developed.

Table 2.1 Characteristics influencing risk in requirements determination (Armour,1993)

Complete	everything the system is required to do is included in the requirements
Correct	every requirement statement must represent something that the system requires
Maintainable	changes needed to the requirements can be achieved easily, completely and consistently
Traceable	the origin of each requirement is clear
Unambiguous	each requirement statement must have only one interpretation
Validatable	every requirement statement must be validatable, either by manual or automated means in a finite cost effective manner
Feasible	each requirement must be achievable within the scope of project resources
Consistent	no subset of individual statements may have conflicts
Precise	each requirement must be stated in manner that is clear and specific to both user and designer
Testable	each requirement must be stated in manner that allows a test case to be developed for it
Understandable	the requirements must be understandable by both users and designers
Verifiable	every requirement statement must be verifiable, either by manual or automated means in a finite cost effective manner

Errors introduce the potential for multiple interpretations, which may cause disagreements between users and designers and result in costly rework, lawsuits or unproductive system. Given the high cost of requirements errors, it is widely acknowledged that effort is needed to determine the causes of errors and how to address them. It is at this level that risk in requirements is addressed and that risk management is introduced as a necessary component to produce quality requirements with the characteristics shown in the above table.

Designers and users view requirements issues from very different perspectives. Users may have a difficult time articulating an accurate, complete, and precise picture of system needs. Errors occur during the requirements phase because the user may not clearly understand system needs and/or may use imprecise or ambiguous terms to describe these needs. Designers may lack the necessary communication skills needed to elicit system needs. The designers may not be sufficiently acquainted with the user domain and therefore unable to determine if the requirements specifications accurately reflect system needs. Users and designers may speak different "languages" and lack a common ground to communicate. The language of the user is usually specific to the domain, while the language of the designer is based on the technology used to attempt to solve user needs. In addition, there may be multiple users with different and incongruent views of the system needs. These users may not be able to visualise how a system will satisfy their needs (Goma, 1983). Furthermore, designers may not be able to represent the system via paper-based requirements in a form that users can understand and relate to their needs. These paper-based requirements are the result of transformation process that may not accurately record the intent of the user (Palmer & Aiken, 1990). Users and designers lack the complete perception needed to ascertain the accuracy of such a transformation. This disparity in understanding must be bridged to effect a successful transfer of user needs to requirements specifications.

Brooks (1987) feels that it is extremely difficult for users to articulate "completely, precisely and correctly", an accurate set of requirements without first iterating through versions of the system. These versions allow users to visualise how the system satisfies their needs and help to simulate any as yet unarticulated needs.

The difficulties described above herald a significant risk that requirements will not reflect a clear and accurate understanding of the problem being addressed. The next section of this chapter discusses the main problems of requirements elicitation in information systems development.

2.3 Requirements determination problems

This section discusses the problems associated with complex and uncertain design domains. The aim is to highlight pertinent features of enterprise information requirements determination as a precursor to being able to give a comprehensive description of the problems that the thesis investigates.

The main difficulties with requirements determination are complexity and uncertainty. Constant change in business and highly dynamic business environment escalate these problems and make them harder to cope with. Different representation methods have been developed to lessen the problems of uncertainty and complexity, by supporting better communication between users and designers. These issues will be discussed in detail in the rest of this section.

2.3.1 Complexity

Complexity normally implies large amounts of information with many parts and many interconnections. Its presence, therefore, suggests difficulty in reasoning or prediction.

Complexity is a structural quality. More complex entities arise out of a combinatoric play upon the simpler entities. The larger and richer the collection of building blocks that is available for construction, the more elaborate are the structures that can be generated. The complexity of a large structure is due to the number of the simpler ones, plus the complexity of interactions between them.

Two types of complexity that are important to design can be identified as state complexity and process complexity. State complexity is associated with describing the state of a design problem, or its representation. The processes by which the world, or

a design problem, moves between these states is associated with process complexity. Both these forms of complexity are dependent on the representation chosen for the entity being described. When describing the complexity of any entity, it is the complexity of the representation of that entity that is being delineated. An absolute measure of the complexity of physical objects, for example, can never be provided. Instead the complexity of some representation of the physical reality has to be described. In the same way a description of the complexity of a process, such as performing a design, can only be given in terms of the complexity of a representation of that process (Williams, 1990). In subsection 2.3.3 the problems related to the representation will be discussed.

Systemic risk is a result of the complexity implicit in all large systems, but particularly in socio-technical systems where people interact closely with the technical system. This sheer complexity, not just in the technology itself but in its interaction with many application environments is likely to cause problems (Angell & Smithson, 1991) in requirements determination of information systems.

2.3.2 Risk and uncertainty

Uncertainty can be thought of as complementary to, or the counterpart of, complexity, in that, in the same way as complexity is used as a measure of amount of information present, uncertainty is used to measure the lack of information.

Just as complexity is associated with having too much information, uncertainty is associated with having too little. The concept of certainty leads to many thorny problems. Indeed much of Western philosophical effort has been directed at establishing what can be known with absolute certainty. For the purpose of design actions the term certainty can be used in a constrained sense, which takes account of the finite nature of processing powers of men and machines (Williams, 1990).

We define requirements uncertainty as the difference between the knowledge already possessed about a problem and knowledge that is needed to derive an acceptable system for the users. Requirements uncertainty can result from among other things:

communication difficulties between user and analyst, inexperienced analysts, technologically naive users, and unstructured tasks supported by the proposed system. The most important forms which requirements uncertainty may take are ambiguity and conflict or inconsistency. Ambiguity exists when requirements statements that are vague may be interpreted by the user in an unexpected manner. Conflict or inconsistency exists when requirements statements which are in conflict with each other may result in confusion, misinterpretation and incorrect design decisions being made by designers. Both these forms can be highlighted when we represent a problem. This is the subject of the next subsection.

Risk may occur during requirements determination because of a lack of certainty. Risk manifests itself through the characteristics of imprecision, conflict, incompleteness or ambiguity in requirements statements used to represent system needs (Sage & Palmer, 1990). Uncertainty is created as to whether requirements statements accurately reflect system needs, and increases the risk of an unsuccessful development effort. Risk is defined within the context of requirements determination as the likelihood that user needs will not be accurately represented in a requirements specification containing such characteristics. Requirements at risk in this context are ones that contain characteristics such as ambiguity and inconsistency (Armour, 1993).

2.3.3 Representation

A representation of a system is an ordered arrangement of symbols that stand for objects and relations in the real world. The way in which symbols can be ordered or built up in a meaningful way is dictated by notations and a grammar. By defining symbols and notations in a way that allows unnecessary information to be suppressed, complexity is reduced. This process of limiting information is referred to as abstraction. It is this abstraction process that yields the capacity to describe real world objects with reduced complexity. The complexity of a system depends heavily on the representation chosen, i.e. the symbols and notation. For instance a computer system may be represented using the notation of block diagrams with symbols for the CPU, interface and memory; alternatively it could be represented using a circuit diagram. The complexity of the latter will be far greater than the former.

The grammar of a representation defines meaningful relations between symbols and notations. The grammar can reduce ambiguity and inconsistency by providing a consistent and understandable representation. The representation itself can reveal inherent conflicts and uncover ambiguities in a way that reduces uncertainty and as a result can provide a design effort with reduced risks. The next section discusses different representation techniques employed in design domains.

2.4 Requirements representation techniques

The previous section described the major difficulties of requirements determination as complexity and uncertainty; difficulties which are intensified by today's business change environment in enterprise design domains. Representation techniques have been developed to facilitate communication between users and designers and overcome the difficulties of information-based design domains. This section offers an overview of existing requirements determination techniques.

Described in this section are the common specification techniques used in requirements analysis. The techniques have been developed, in part, to address the issues discussed above. Formal specifications are an attempt to represent requirements in a manner that facilitates common understanding as well as reducing errors such as ambiguity and inconsistency. Requirements documents created by designers and presented to users for the purposes of requirements definition traditionally include (Carey & Mason, 1983): textual lists of requirements, an interpretive model of the proposed system and a working model of the proposed system. The first two of the three techniques attempt to represent the proposed system behaviour through abstraction. Although each of the techniques brings specific benefits to requirements specification, abstract representation can present significant difficulties. This leads to the third method, prototyping for specification of user requirements.

Table 2.2 lists commonly used requirements representation techniques and places them into the above mentioned categories.

Table 2.2 Requirements representation Techniques (Armour, 1993)

Text-based requirements	List of requirements Narrative English descriptions
Interpretive models	Structured Requirements Definitions (SRD) Structured Analysis Object Oriented Analysis (Object model) Finite State Machines Statecharts Decision Tables and Decision Trees Entity-Relation Diagram Data Models
Working model (traditional prototypes)	Storyboard prototypes Paper-based prototypes Breadboard prototypes Simulation prototypes Skeleton prototypes Throw-away prototypes Evolutionary prototypes Executable requirements prototypes

One of the major difficulties of requirements definition is communication. Users and designers do not generally communicate on even terms (Palmer, 1988). A "semantic gap" (De Brabander & Thiers, 1984) may exist between user and developer. The inability of text-based and interpretive methods to bridge completely this communication gap leads to the use of a third method- prototyping. When requirements are poorly understood they are likely to change during the development life cycle, resulting in a final product that does not meet user expectations. Prototyping attempts to provide a common ground of understanding between users and designers by presenting users with a relatively realistic model of how the system will appear and/or behave. In arriving at a common understanding concerning the acceptability of a proposed system, it has been shown that users and designers benefit from viewing examples of, or having practical experience, with a working representation of the system (Boar, 1984; Davis, 1990b).

This section first reviews several analysis and representation methods for requirements analysis, and presents a rationale for prototyping during the requirements determination phase of the information systems development.

2.4.1 Textual list of requirements

Text-based requirements specifications are traditionally used to list the requirements which the system must meet. Text-based specifications, usually in the form of unstructured English, are a communication medium very familiar to the average user. In fact, even if another form of requirements representation is officially used, such as an interpretive or a working model of the system, most requirements are still presented, however informally, in a text format at some time during the requirements determination phase. When deriving initial requirements from users, text-based specifications provide a communication advantage over an interpretive model of the requirements specifications. However, because of the inherent imprecision and ambiguity of natural language, text-based specifications have the potential of introducing errors into the requirements definition process.

Text-based requirements tend to be lengthy and difficult to read because they are psychologically distant (Carey & Mason, 1983) from what the users will eventually receive as a software system. Text-based requirements have been known to be as extensive as five thousand pages (Davis, 1990b). When developers are assigned to analyze system requirements and develop a written specification document reflecting those requirements, users often find great difficulty in understanding the documents and still miss errors embedded in requirements statements. Inconsistency, conflict, and redundancy often insinuate themselves into a large text-based document. Many system features are difficult to represent using a text-based description, for example, clearly defining how a graphical user interface will look and behave using just text is nearly impossible.

2.4.2 Interpretive models

Interpretive models are graphical representation of what system will do. Examples of graphical interpretive techniques include: Structured analysis (SA) (DeMarco, 1979;

Yourdon, 1989), object oriented analysis (Coad & Yourdon, 1990; Coad & Yourdon, 1991) and object model (Rumbaugh, Blaha, Premerlani, Eddy, & Lorenson, 1991), state transition diagram, and statechart (Harel, 1987). Also considered within this category are formal specification languages. Interpretive techniques such as graphical representations of requirements and specification languages attempt to provide a common and precise language for requirements specification. Interpretive techniques try to reduce the inherent imprecision and ambiguity that exist in natural language (Davis, 1990b). If a technique is computer automated, the computer can provide support to trace inconsistencies, redundancies, and ambiguities.

The format and syntax of interpretive models may be foreign to the user, making understanding and evaluation difficult. As the model becomes larger, it becomes even more difficult for users to comprehend the representation. Our personal experience in trying to review page after page of a sizeable ER model has convinced us that for large systems, interpretive techniques present difficulties in understanding requirements representations. One problem with this modelling technique is how to decide where to begin reading the chart. While interpretive models have been beneficial, they still present requirements in a form to which users find difficult to relate. If needs are not understood, a precise requirements language is of little use (Taylor & Standish, 1982).

2.4.3 Working models or prototypes

In the late 1970s and early 1980s, the idea of prototyping emerged as a distinct software development technique. Originally the concept of prototyping in software engineering was borrowed from other engineering disciplines. In manufacturing industry the scientific method to problem solving has been used in applied engineering problems, in the form of the prototyping approach. A physical scientific approach presents a simple method for how scientists solve problems. The scientific approach to problem solving is an old and well-established technique which is widely used in the physical sciences. The following figure (figure 2.2) shows the steps involved. The objective is to provide a better understanding to a problem-oriented approach in physical science.

In this method the problem is *to formulate a hypothesis* which gives an accurate and

consistent description of the observed behaviour of the system being studied. The solution is then *the theoretical model* used to capture and predict the behaviour of the system (Maude & Willis, 1991). By using this method, scientists are able to assess the validity of the problem formulation prior to proposing the final solution. Through the cycle of hypothesis-experiment-verification and revision a better convergence between the prediction of the model (solution) and the behaviour of the system being modelled (problem) can be achieved. This cycle of experimentation of the validity of the hypothesis formulation is the key idea in physical scientific approach to problem solving.

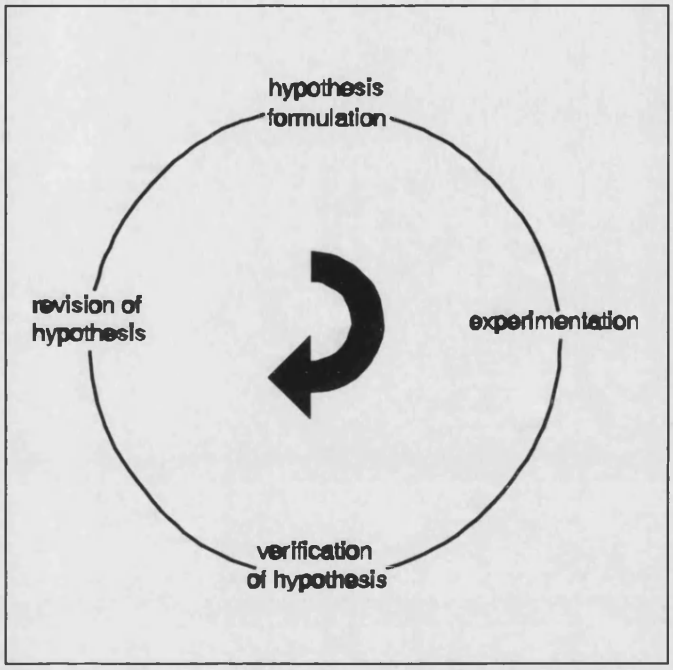


Figure 2.2 A physical scientific method for problem solving

In the prototyping approach, based on a scientific model, instead of preliminary formation of the hypothesis, a scaled working model is built in order to discover the problem involved in manufacture; and then the cycle of experimentation of this model will be exercised until validation of the solution to the problem is achieved. By constructing a scaled-down prototype version of the envisaged computerised information system, the analyst can present not only a model of the organisation's conceptual information structure, but also a model of how the information will be processed after the computerised information system has been constructed and installed.

Unfortunately until now, there has been no real agreement on the definition and categorisation of a prototype within the context of software engineering. The different expectations and the various ways in which prototypes can be used in systems development have led to many attempts at classifying prototyping. The most popular

classifications include (Maude & Willis, 1991):

- data-driven prototyping (Appleton, 1983)
- exploratory/experimental/evolutionary prototyping (Floyd, 1987)
- horizontal/vertical prototyping (Floyd, 1987)
- mocked-up/breadboard
- throw-it-away/incremental/evolutionary
- exploratory/experimental/organisational (Dearnley & Mayhew, 1983)
- early/middle/late prototyping
- cooperative prototyping (Bodker & Gronbaek, 1991)
- explanatory/exploratory/experimental/evolutionary (Maude & Willis, 1991)

The Shorter Oxford dictionary defines the prototypes as follows:

"a prototype is 'the first or primary type of anything; a pattern, model, standard, archetype' being derived from Greek words *protos* meaning *first* and *typos* meaning a *type* [Shorter Oxford Dictionary]."

In most reports, the process of problem solving in building a prototype has been named "a learning process". First-of-its-type is a pilot system, so that prototyping is the initial attempt to produce the system, purely with the intention of learning how to do the job properly (Brooks, 1975). Prototypes are working models of the system and present users with a realistic view of how the system will behave. This technique is discussed in detail in the next section.

2.5 Requirements prototyping process

According to the scientific problem solving approach, problem formulation is concerned with eliciting user requirements for an information system and the representation of the solution is concerned with stating the specification and properties of the system precisely and unambiguously (Hekmatpour & Ince, 1986). Every representation of the solution to the problem needs to be validated: ensuring that the final developed system built around the specification can meet its user requirements. In seeking for a better fit solution, the concept of prototyping uses the cycle of prototype-exercise-verify-revision to allow a model of the system behaviour (a representation of the solution) to

evolve toward final solution. Figure 2.3 represents the requirements prototyping process.

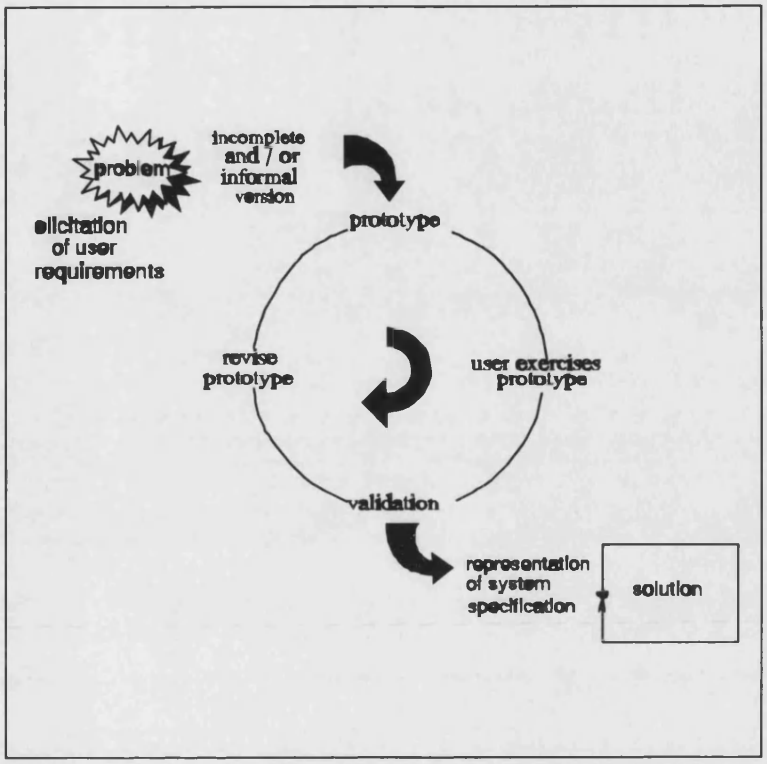


Figure 2.3 Prototyping approach to problem solving

Developers elicit initial requirements information from users, transform these original requirements into an informal specification, and then develop a prototype based on the informal specification. The prototype is then evaluated by users and the feedback is used to refine and to formalize the requirements specification. This iterative process continues until users and designers feel satisfied that the specification adequately defines system needs. The output of the process is a refined set of requirements specifications as shown in figure 2.3 and discussed below. The final requirements specification acts as input into the software design effort. It should be noted that the specific goal of a requirements prototyping activity is the refinement of a requirements set that will guide the future design effort. These prototypes can be used or enhanced during a later phase of development.

Requirements prototyping increases user awareness of the developing system (Sage & Palmer, 1990) by allowing users to view how a representation of the system will function before the formal requirements specification is defined. In contrast with paper or written specifications which can be very difficult for users to understand and to relate to the proposed system functions, prototypes allow a diverse set of users to discover

how the system will look and operate. If users cannot understand the requirements specification presented to them by designers, they will not be able to determine whether the resulting system will address system needs. Prototyping helps eliminate this uncertainty by presenting refined requirements information in a form which both users and designers understand and can communicate. Requirements prototyping represents an attempt to narrow the communication gap between users and designers. Communication breakdowns between users and designers can be discovered through the use of requirements prototyping (Armour, 1993).

The result of the requirements prototyping process is the identification of ambiguities, omissions, and inconsistencies in the specifications (Goma & Scott, 1981). The benefits, when compared with other formats of communication, include reduction of risk by ensuring that requirements specifications are more complete, correct, consistent, and unambiguous.

Since cost, time, and maintaining user interest will probably be important criteria driving the requirements prototyping process, it is important that the prototype be developed as quickly as possible. This requisite is often referred to as rapid prototyping. For a prototype to be effective, it must be developed quickly so that it may be evaluated in the requirements analysis phase of the system development life cycle (Goma, 1983; Goma & Scott, 1981; Sage & Palmer, 1990). Users will grow impatient if the requirements definition is stretched out over a long period of time. If concrete progress is not demonstrated to users, they may lose confidence with the process, creating unneeded tension and difficulties between users and designers.

Prototyping is a useful technique to address the requirements determination problems mentioned earlier because:

- Since prototyping provides a closer representation of the system than the other forms of requirements representation, it provides a simpler and less ambiguous means of communication between users and developers. When requirements are reflected through features in a prototype, conflicts in requirements statements become more apparent. Prototypes allow both user

and developer to view requirements in a clear, less ambiguous form. Conflicts can no longer hide in ambiguities of the English language or the arcane vocabulary of computer terms.

• Prototypes allow alternative requirements to be tested and represented. When given sets of requirements are in conflict, developers may be unsure which set of requirements to implement. Prototyping allows the effect of alternatives to be demonstrated and explored.

• Users get to see the system or subsets of the system as they appear, reducing the uncertainty that the developer has misinterpreted user comments.

• Subsystems, such as the human-computer interface, which cannot be accurately represented with written requirements may be evaluated.

As prototypes are a cheap, flexible and simplified working model of a system with the potential of use in an operational environment, conventional approaches have benefited from building prototypes in different phases of software development: for exploration of user requirements in analysis phase or experimentation of the different solutions in design phase or as a testbed during implementation phase. Because in conventional approaches the development process is organised in a sequential order, there is little room for further modification on previously frozen stages. As a result some researchers tried to incorporate the prototyping technique into the conventional approaches in order to address some of their deficiencies. At any stage where we need a cycle of user experimentation in order to provide a better understanding of the problem, analysts can use the prototyping technique and benefit from user participation towards finding the solution. Many organisations that adopt the life cycle model also use prototyping (Boehm, 1981). However, the central importance of conventional approaches as a support model to software engineering has always been widely acknowledged by many developers (Musa, 1983; Riddle, 1984).

Despite the fact that the proponents of conventional methods believe that prototyping is simply different practice that can be plugged into existing phases of the life cycle, many researchers have mentioned that, by retaining the life cycle model we inherit its shortcomings, which limit the benefits that can be realized from new paradigms (Agresti, 1986). The conventional approaches are strongly influenced by a reductionist pattern of problem solving. Conventional approaches follow the goal of developing systems by achieving sub-goals through a phased-transition prescriptive sequence. Although it provides a systemic analytical approach to problem solving, its lack of synthesis is a fundamental weakness. Agresti (1986) mentioned this issue as an imbalance between analysis and synthesis. He mentioned as an example early experience of playing a game, even though not following all the rules, is a more effective route to mastery than continuing to analyze the rules. So even by incorporating prototyping in different phases of conventional approaches, we are still fixed in the realm of static representation of specification and design. Conventional approaches are associated with phased control points and intermediate products through an analytical process, so they neither acknowledge nor exploit the balance between analysis and synthesis.

In this respect, prototyping is fundamentally different from the conventional approaches. It adds the elements of synthesis to the analytical approach to problem solving. Prototypes are built to be changed, and development is iterative not linear (Angell & Smithson, 1991). The basic idea of prototyping as a completely independent approach to software development derives from the fact that it is a working model, so it can evolve into a complete system. This idea introduced a new concept in information systems development named an "evolutionary development" approach. This is in complete antithesis to traditional approaches of information systems development methods.

2.6 Evolutionary development approach

Evolutionary development is a subset of the prototyping approach, in the sense that systems are designed to be changed. The main difference between the evolutionary

development approach and traditional prototyping is that evolutionary systems evolve in use instead of experimentation. In the use environment, uncertainty is the result of the inherent turbulence and dynamism of the environment.

Proponents of the evolutionary development strategy argue that information systems once installed evolve steadily, invalidating their original requirements (Brittan, 1980; Naumman & Jenkins, 1982). The purpose of the evolutionary approach is to introduce the system into an organisation gradually, allowing it to adapt to the inevitable changes that take place within an organisation as a result of using the system (Rzevski, 1984). Evolutionary development is by far one of the most powerful ways of coping with change. This approach requires the system to be designed in such a way that it can cope with change *during* and *after* development. A design practice that does not take the possibility of change into account can lead to severe problems; this is illustrated by the following revealing extract from a description of the effect of organisational change on an existing information system (Alter, 1980).

"... systems were strained badly or died as the result of corporative reorganisation...An old version of a planning model was abandoned as a result of a reorganisation, only to have its basic logic restructured years later...The conceptual design problem here is building systems that are truly flexible..."

In evolutionary prototyping a system grows and evolves gradually (Nosek, 1984). For this reason the first prototype usually does not implement the whole application. Instead, enough development is carried out to enable the user to carry out one or more tasks completely. Once more is known about these tasks and how they may affect others, more parts of the system are designed and implemented so that a larger section of the task domain may be covered. This allows a low-risk, continuous and gradual development while the system is undergoing use.

Evolutionary prototyping starts with gradual development of **requirements prototypes** and then allows the prototype to evolve and be adapted in the use environment. At some point in time the final prototype is eventually transformed into a system.

Depending on how well the system design has survived the evolution process the final prototype may serve as the production version, or a complete redesign might be necessary to facilitate maintenance. Obviously the availability of appropriate tools is vital. To cut down the redesign effort and be able to develop large systems, a highly modular design which can cope with extension and contradiction (Parnas, 1979) will prove highly effective (Hekmatpour & Ince, 1986).

In the evolutionary environment, although the development process cannot freeze the turbulent conditions, it still can benefit from the ability of the prototyping approach to cope with uncertainty and change in requirement analysis. The evolutionary development approach benefits from prototyping in all stages of development, starting with a limited definition of the problem while trying to provide an environment which can continuously improve. Therefore, the approach benefits from the prototyping for requirements determination at the early stages of the project and then shifts to a stable change condition to improve prototype system. In this shift it tries continuously to improve the system specification by introducing better fit prototypes.

Evolutionary development is a continuous improvement environment where prototypes evolve in use instead of experimentation. As conventional life cycle methods have mostly benefited from prototyping in the early stages of development, evolutionary environment by starting from the limited definition of the problem, tries to provide a continuous improvement environment. Having a stable strategy for the problem boundaries and pinpointing the target system by denying the feasibility of any final specification (Angell & Smithson, 1991) is the main contribution of the evolutionary approach.

The idea behind the evolutionary development rests upon the understanding that it is not necessary to have an articulated understanding about the real world to be able to design artifacts. Human beings have the capability to work in the real world using tacit understanding and by repetition of processes can gain experience and adapt to the situation. As Floyd (1987) averred "we ought to think of design as redesign" not as a "one-shot" process. A continuous improvement approach reduces uncertainty by

demonstrating the growing shape of the new system continuously.

A continuous improvement feature also reduces uncertainty by demonstrating the growing shape of the new system. This feature acts as a concrete vehicle for user involvement (Angell & Smithson, 1991), and users can develop more easily their conceptual understanding about the problem. However, reliance is still placed on the solidity of the users' ideas and any misunderstandings and omissions should be highlighted and clarified.

The continuous improvement environment in evolutionary development is a viable strategy for information systems development in companies whose goals are relatively stable and predictable. But changes in the real world business are dynamic and sometimes uncontrollable. Those companies which need to compete in their industry and survive with everchanging and highly turbulent goals cannot totally benefit from this approach. This condition will be intensified by factors such as ill-defining the problem domain and mindlessly adopting technological advances. Most companies now have manifold goals and objectives as their corporate strategy. They are shifting from the "OR" world of seeking for one of their goals at each time upward to the "AND" world of having all of them at the same time.

Although the evolutionary development provides a promising approach in a business change environment, the complexity of large systems and the inherent difficulties with requirements prototyping imposes limits on evolutionary effort. From a technical perspective, evolutionary prototyping lacks the controls needed for project management and for reliable outcome measures. In addition, there is a lack of clear rules determining when the prototyping process has reached its goals. To put it simply: the user's appetite for change could continuously grow and there is no guarantee that the changes made are worth their costs.

From a social perspective, the high tech nature of evolutionary prototyping is of concern for two reasons. First, there is the danger that communication and learning are influenced by the ideas and values underlying the latest technological fashion and not

by genuine social concerns. Second, and more fundamentally, an inescapable limitation of prototyping is that it treats information systems as technical systems which can be discontinued without further consequence if deemed deficient. This is fallacious (Hirschheim, Klein, & Lyytinen, 1995). The next section explains the difficulties associated with evolutionary development approach when applied in complex and turbulent business change environment.

2.7 Current difficulties with evolutionary prototyping approach

Some of the problems associated with evolutionary development are rooted in the difficulties of using requirements prototyping in determining users' needs. Others, specifically the management of the evolutionary process, are more specific to the approach itself. The business change environment escalates the problems of this approach when applied in large systems, and at the same time an evolutionary approach would seem to be the only effective option in development of large systems in business change environment. The aim of this thesis is to investigate the major difficulties in the evolutionary development approach, in order to enhance it and to enable the approach to be used more efficiently in practice.

2.7.1 Problems with managing the implementation process

The most important negative side of using evolutionary development is that the development process is more difficult to manage and control than conventional approaches. Because of the issue of managing the implementation process in evolutionary development, sometimes the underlying benefit of using this approach, i.e. its simplicity, is completely lost where systems are extremely large and complex.

The difficulty of managing evolutionary development projects often stems from the lack of a prescription as to how much of an organisation's information requirements to include in each iteration of the prototype (Alavi, 1984; Andrews, 1983; Dennis, *et al.*, 1987; Mahmood, 1987; Pliskin & Shoval, 1987) and knowing where to start and where to go next (Carpenter, 1992). Without a blueprint of how to manage a prototype the

risk of *ad hoc* activities during the implementation process would be great (Albadvi & Backhouse, 1994a). As Angell and Smithson (1991) described:

"... evolutionary approach explicitly addresses the problems of uncertainty and change in requirements analysis. It permits much scope for user involvement by allowing time for users to learn the evolving system. However, it may suffer through its own inherent insecurity and uncertainty, in that developers and users cannot know how long the particular version will survive before it is changed. Even more troublesome is the case where the lack of planning has meant that there is insufficient flexibility (or upgrade path) for the system to evolve..."

Project planning is one of the most important, and yet most difficult areas of project management: the old axiom that "if you can't plan it, you can't do it" haunts project managers (Angell & Smithson, 1991). Any effort towards development of information systems introduces new practices to the organisations, changes in the way people are used to do their tasks. In this sense information systems development means inducing more changes to the business change environment. Therefore, planning the evolutionary process has two aspects: planning how to manage and cope with an everchanging environment and how to induce more changes (new information systems) to this environment. The two sided nature of project planning is one of the most important obstacles when using evolutionary development approach in large systems. This obstacle raises the issue of conceptual clarification in development of prototypes. Conceptual clarification should aim at formulating a theoretical foundation and at assessing more finely the implications of different versions of prototyping. The problem of conceptual clarification has led the research to focus more closely on the other difficulties of evolutionary prototyping approach as discussed in the following subsections.

2.7.2 Lack of a theoretical foundation for analysis of requirements statements

Using the evolutionary development approach without a model for assessing the sustainability of the problem formulation can lead to unpredictable dynamic changes. A prototype has been defined as a model of a system that is yet to be built. However,

defining a prototype does not explain how to go about building one (Maude & Willis, 1991). Providing a model to support development of prototypes is a crucial issue in using this approach. Apparently developers who use this approach implicitly attempt to conceptualise the intermediate targets as a model for their work. Such an attempt tries to explain how to develop a prototype, but obviously never really succeeds. All that can be realistically done is to explain the short term movements. What developers need is a strategy that points the way: a model which can provide some guidelines as to where to start, and where to go next, given the context of organisational goals. The lack of a support model for planning a prototype is another important shortcoming of prototyping. Without a clear strategy there is always a risk of being confounded by the slippery slope of dynamic changes, where insecurity and uncertainty will be the result.

In reality the social nature of information systems, where different groups may have very different requirements and expectations, means that it is impossible to specify requirements objectively (Angell & Smithson, 1991). This and the practice of regarding the machine as central to information causes the neglect of qualitative issues, the natural result of the excessive emphasis on the technical requirements. Without a support model with the ability to view the information system as a socio-technical system and to reveal the fundamental place of people, evolutionary development may lead explicitly to an implementation-oriented environment. This environment has the perspective of beginning with the formal system instead of with the context and the purpose of the information. This process will lead the implementation effort to focus on those system specifications which are more amenable to software techniques. Because of this weakness some researchers are advising the use of this method only for small to medium size systems (Angell & Smithson, 1991; Hekmatpour & Ince, 1986; Land, 1982), where the scope of the problem, number of users and the list of goals and expectations are reasonably manageable. Without a supportive conceptual model, prototyping a large scale system through dividing it into subsystems may result in diverging systems, difficult to integrate later on. The technical-orientation of prototyping suggests a danger that communication and learning may be biased by technological advances not by social concerns. A socially constructed conceptual model can direct the effects of prototyping in the minds of the affected users. One stern

institutional safeguard resulted from such a model is that the role of organisational actors within the informal system should not be forgotten in the enthusiasm for the high tech gadgetry in prototyping projects. It is important that users gain access to adequate means to support their design understanding and evaluation of prototypes. The following subsection addresses this issue in detail.

2.7.3 Portray the requirements together with the prototype

The lack of techniques to view effectively original requirements during prototype presentation can result in a less than satisfactory evaluation, because of information loss and misunderstanding. This is another major difficulty with requirements prototyping employed in the evolutionary development approach (Armour, 1993), which is the subject of this subsection.

Information loss and a prematurely bounded system representation

The original requirements information from which a prototype is derived may come from variety of sources and much of it may not be clearly understood by either prototype developers or individual users. To prototype requirements, the developer needs to prespecify the requirements into an informal specification (Balzer, T.E. Cheatham, & Green, 1983). Initially captured requirements information needs to be refined and organized into a form that prototype developers can use to develop a prototype. But information is lost in this process. Requirements uncertainty becomes requirements certainty. This process formalizes or narrows requirements at a very early stage of the development. It is almost certainly an imperfect transformation. The requirements prototype development is based on this more specifically stated set of transformed requirements. During prototype evaluation users may not possess a clear understanding of, or may have difficulty identifying, how the prototype does not address needs.

When users are presented with a prototype for evaluation, they view the prototype as the system, even though the prototype is used as a tool to elicit refined requirements from them. Scharer (1983) cautions that during requirements prototyping, users must be careful not to allow a prototype that may look like a final system to preclude

consideration of new alternatives or even to alter the basic system approach. Users must not be cognitively bounded by the prototype representation. The prototype was probably derived from an imperfect transformation, and the user may not recognise needs incorrectly transformed. It is the prototype which is presented, not the originally captured requirements information necessary to discover prototype correctness and completeness.

Original requirements information is transformed into refined requirements that serve as input to the requirements prototyping process. The prototype based on this refined information is presented to the user for evaluation. Users evaluate the prototype that reflects these refined requirements. If the original requirements were incorrectly dropped or incorrectly interpreted during the initial refinements process, the user may not have direct access to the information, and may incorrectly evaluate the prototype.

Information misunderstanding

Information needed to develop the requirements prototype is of necessity in a structured format and may not be in its original, perhaps domain specific, form. The information may not be as understandable to the users and therefore may not provide an accurate representation of needs against which to evaluate the prototype. Davis (1982) found that information needs elicited or experienced most recently are given more weight by users than needs discovered or elicited earlier in requirements gathering.

Collected requirements information may be comprised of hundreds or thousands of documents, collected over a long time frame. Information may not be known or may be forgotten by the user. Problems with requirements information may be difficult to determine because the information needed to help determine whether requirements are incomplete, inconsistent, or ambiguous may be scattered across the multitude of documents. Psychological studies involving memory have firmly established that recall in memory tasks is much more difficult than recognition. If prototype evaluation is performed without reasonably easy access to requirements information, users may have an incomplete or inaccurate view of need. For users to evaluate accurately a prototype, they may need access to the original requirements information used by developers to

generate specific prototype features.

Although prototyping is a method to bridge the communication gap between user and developer, it still is more of a developer language than a user language. The developer understands software systems and is comfortable with their concepts. A prototype is a software system which the developer has created from an interpretation of user needs. Users may not be comfortable or completely understand the software system and therefore may be unable to determine whether it meets their needs. The prototype may be a conglomeration of multiple and perhaps competing user needs. A prototype may have many different users with differing requirements. Users may not comprehend which functions of the prototype address specific needs or they may misinterpret prototype features, since they cannot clearly map prototype features to specific needs. Conversely, developers may not understand requirements information well enough to reflect it accurately in the prototype. Users and developers do not always share a common information base. Users view a system from two sub-models: the system itself and their goals in relation to that system (Woodhead, 1990). Without accurate, understandable, and complete information supporting the goals for the system, it will be difficult to relate correctly these needs to the system.

Users may not be able to visualise how modified requirements can impact the system. Bostrom (1989) found that although a prototyping approach can help the requirements analysis process, there are still communication problems between users and developers that are time consuming and frustrating. If users are unable to express needs or understand how the prototype relates to these needs, the evaluation will result in incorrect or incomplete requirements specifications.

Finally, we can view requirements prototyping evaluation from the perspective of a group decision process (Fields, 1991; Kraemer & King, 1988), in which users and developers brainstorm (Boar, 1984) or discuss ideas on how the prototype can be modified to meet user needs. During the requirements phase it is important that users and developers come to a consensus on the requirements used to develop the system (Charette, 1986). Apart from organisational politics, the information and

communication problems described earlier alone may prevent such a consensus from ever being reached.

The above problems highlight the need for requirements traceability when presenting prototypes during evaluation sessions. Without the ability to portray original requirements together with prototypes and efficiently trace back each prototype to its original requirements, evolving systems may not lead to desirable integrity in the use environment. Establishing a platform for clear understandings for users and designers using an appropriate conceptual model would open up new opportunities to implement some checks and balances against introducing obstacles and biases to rational communication. This can be done by tying ideas and abstract concepts to concrete circumstances in the workplace. Designers can support the evaluation and revision of evolutionary prototypes by tracing them back to their associated conceptual models which users have previously agreed upon. The ideal is to arrive at an enterprise-wide conceptual model as a good support environment that facilitates high quality design understanding.

2.7.4 Lack of an effective technique to assist in the discovery of requirements features and in prototyping subsections of a large system

It may not be feasible to perform requirements prototyping on all features described in the entire set of requirements statements in large systems. Determinations of which requirements or groups of requirements need to be prototyped should be made. A trade-off occurs between resources and time available to perform prototyping and the degree of robustness and completeness that can be achieved in the prototype and then reflected in the requirements specifications. The areas of greatest uncertainty (hence greatest risk) should have highest priority when the prototyping effort has to be limited in its scope. To prototype successfully a proposed subsystem, or feature and review it with users for the purpose of requirements refinements, all the requirements relating to these features must be collected, organised and analyzed for their complexity and uncertainty. If a large number of requirements is present, manual classification can be at best unwieldy and time consuming, and at worst impossible.

Simply collecting and grouping requirements poses a significant challenge to the identification of requirements to prototype. Without a reliable method to select requirements from a large set and cluster them into a group, the effectiveness of the identification of requirements criteria to be prototyped is limited. A technique is needed to identify and cluster together all the requirements presenting similar features, so that robust prototypes can be developed based on all requirements containing references to these features. Several methods currently being used for the identification of requirements with similar features are discussed below (Armour, 1993).

Manual review

Designers or users can manually review the requirements to identify and group those with specific characteristics. An individual or group of individuals reviews each requirement and compares it with other requirements to determine whether they belong together and at the same time whether they possess characteristics that make them viable candidates for prototyping. When people review or compare large sets of information, it is likely that they miss critical pieces of information or factors because humans are unable to perform comparisons of large sets of information. This is described as inexpert indexing and can lead to classification faults that include (Armour, 1993):

- the clustering of statements around concepts based on incorrect terms;
- the failure to recognize a statement containing a concept which needs to be clustered;
- and the clustering of a statement that should be ignored.

Additionally, questions also arise from inconsistencies generated because different individuals are performing the classification. Therefore, for a large requirements set manual identification and grouping of requirements is not a viable method. We are forced to look for automated approaches for requirements identification and grouping.

Automated keyword search

An automated search of the requirements set by keyword would produce identification

and grouping of requirements based on the requirements containing specific keywords. A keyword based indexing approach searches a set of text-based documents using keywords as characteristic indicators of text terms. An automated search will be faster and more accurate in finding statements in situations that would have been too unwieldy or time consuming using a manual approach. Many information retrieval systems employ this form of indexing.

Automated keyword search also has its limitations. At the simplest level, a keyword search approach is limited by the fact that it identifies and matches requirements statements based only on exact comparison of each keyword to a word in a requirement statement. The search will miss information that is related through synonyms or similar phrases and most important through meanings and the semantics of words. It does not have the robustness needed to identify statements related by conceptual similarity other than through a direct word match.

Hypertext search

Creating a hypertext based set of requirement statements provides a search model that links related requirements together via associative linking (Smith, 1991; Smith, 1993). A hypertext based requirements document allows users to explore related requirements by navigating the requirements through the links between statements, however, the related concepts in each requirement still needs to be identified first, leading to the limitations described in manual and keyword search approaches.

The inherent limitations of the manual, keyword, and hypertext methods lead to difficulties in being able to examine thoroughly a requirements set. With large requirements sets, manual review is not only inefficient, but human cognitive limitations make it nearly impossible to do properly. Automated techniques, such as keyword searches, address these issues but are limited to exact pattern match identification. A hypertext structure provides associative links between related requirements, but there is still the issue of how the links are generated. Any proposed conceptual modelling to answer the shortcomings of evolutionary prototyping mentioned earlier needs to have the capability of clustering together the related requirements and of reducing problem

complexity.

2.8 Problems statement and overview of the research approach

While the evolutionary development approach provides a promising development method in a business change environment, the complexity of large systems and the inherent difficulties with requirements prototyping imposes limits on evolutionary effort. The above mentioned constraints place bounds on successful system development. One difficulty addressed so far is how to support an evolutionary development approach to trace back evaluation of prototypes based on accurate information, to identify viable requirement candidates for prototyping and to cluster effectively the actual requirements for identification. This difficulty explicitly addresses the state complexity problem of design domains: how to represent the state of a design problem. As described in subsection 2.3.1 there is another form of complexity, namely process complexity: the process by which a design representation moves between its states. The complexity of the process of developing evolutionary systems and of creating a design are yet further difficulties with an evolutionary development approach which need to be addressed. Both process and state complexities should be considered simultaneously when proposing any solution to those difficulties.

A change approach for managing the development of information systems seems to be a suitable direction to resolve the problem of process complexity. To achieve acceptance in a change process, continuous interaction among all parties is critical. Through interaction, characteristics of the object system emerge and become legitimised through continuous modification. Systems cannot be designed in the usual sense, but emerge through social interaction. The mechanism of evolutionary learning from interaction with partial implementations of object system is the way technology becomes embedded into the social perception and change process. Hence, relaxing the problem of process complexity requires further support to resolve the problem of state complexity. As mentioned earlier, state complexity can be tackled by focusing on conceptual modelling or language modelling. The goal is institutionalising an ideal speech situation which in turn validates an object system and modes of design and

implementation by emphasizing social learning during evolutionary systems development.

A conceptual modelling method needs to be advanced in order to support evolutionary development during the requirements prototyping phase. The support model will enable developers to lay down a development strategy when dealing with large and dynamic systems in business change environments. A model which pinpoints the way to develop requirements prototypes is critical in developing a rational prototyping strategy. The modelling method itself needs to have the following characteristics:

• The modelling method should be able to target the deep analysis of patterns of behaviour of dynamic system under study. By doing this and by precluding any procedural analyses which are most likely to be the subject of change, it would be possible to provide supportive models for evolutionary development with relative stability. The support model will be a base model for prototyping efforts during requirements prototyping and also a reference model during evaluation of prototypes.

• Before a prototype is developed, the original requirements information is always conceptually transformed into a refined form that is used as the specification for the prototype. The prototype specification is a subset of the original information. Consequently, the prototype is developed on a limited view of the original requirements. Users then evaluate the prototype against a set of requirements specifications that may not only be incomplete and incorrect, but in a form which users may have trouble comprehending. These factors increase the difficulty of correctly evaluating whether prototypes address system needs. The prototypes so developed may drive the user to an incorrect requirements specification, in part owing to incorrect transformations and lack of understandable requirements information with which to evaluate the prototype. The modelling method needs to develop reference models as a refined form of requirements information for evaluation of prototypes. The prototype specifications are derived from this model and each prototype will be evaluated against the reference model rather than against its specification. The model needs to be understandable and modifiable by users and be based on their domain language. The modelling method

should provide transparency between requirements statements and working prototypes.

• Requirements at risk are ones that contain characteristics such as ambiguity and inconsistency. The modelling method needs to address these characteristics and to highlight any ambiguity and inconsistency. This provides a viable criterion for identifying requirements for prototyping and targeting requirements with greatest risk or uncertainty. A failure to identify systematically these requirements characteristics can result in inherently high risk requirements being left out of the prototyping effort.

• The modelling method needs to be furnished in a way which can effectively group or cluster requirements based on invariant features of requirements objects. The individual requirements statements that reflect features to prototype may be scattered throughout the maze of requirements objects. Difficulties arise when users and designers try to abstract, group, and analyze statements that describe common features. It then becomes problematic to find and identify requirements that benefit from a prototyping effort. With large numbers of requirements, a clustering method can deliver a clustered model of the design domain. Each cluster of the model is reflecting a prototypeable cluster of the system. The clustered model should have the capability to be developed in multi-user multi-analyst environment and to integrate the final system.

2.9 Conclusion

In this chapter the basic features of the design problem have been outlined with an emphasis on requirements determination problems. Two generic problems of design, the dual problems of complexity and uncertainty, have been recognised as the most important hurdles toward final success in large information systems development. Complexity and uncertainty will be increased in today's business change environment, which demands greater amounts of information exchange. We explained that developers are trying to tackle these problems by introducing more efficient requirements representation techniques, amongst which requirements prototyping seems to be the best way to deal with change. The evolutionary development approach was discussed as an approach to information systems development which has the potential

to cope with changes in information requirements. This approach, by using requirements prototyping in the early stages of development and then by continuously improving prototypes, has the potential to offer a dynamic systems development suitable for the business change environment. The principal strengths of prototyping are that it (Hirschheim, Klein, & Lyytinen, 1995):

1) sustains the motivation of users to participate in system development thereby providing the most reliable information on requirements

2) overcomes some of the rigidity of the conventional life cycle model

3) allows the determining and validating of system specifications by conducting experiments

4) supports human interaction, sense-making and the creation of new meanings.

In section 2.7 we focused on difficulties when using the evolutionary development approach in large, dynamic and information-based business environments. Therefore, in order to develop large scale evolutionary information systems, the focus of the research should be on two issues:

1) creating a method for managing the development process of large systems which, while inducing change to the environment by developing new systems, can also cope with existing changes in design environment

2) creating a complementary method that gives designers a conceptual model which is relatively stable and risk sensitive, and has the facilities of requirements traceability and requirements clustering.

CHAPTER 3

A theoretical framework for evolutionary development

Chapter Overview

For many years we bandied about terms like prototyping, rapid application development, evolutionary development but without having a context for them. A study of related literature provided some insights into the problems of the evolutionary development approach. Two critiques of the principles of evolutionary development which underpin this conceptual practice have been discussed. We have argued that managing the implementation process is problematic in the evolutionary development approach, especially in regard to the changing business environments. The lack of a supporting conceptual model as a frame of reference is also another major shortcoming of the approach. The central theme of this chapter focuses on a proposal for a new theoretical framework for information systems design and development. The new theoretical framework is based on planned organisational change theory and semiotic theory in response to these two shortcomings. Section 3.2 begins with the theoretical perspective of change and persistence. After defining different orders of change, it discusses process theories of change in relation to information systems development. In section 3.3, Lewin's three-phased change process model - unfreezing, moving and refreezing - is suggested as a model for managing implementation process in evolutionary development. Section 3.4 presents the main pillars of semiotic theory. This theory focuses on the users' own interpretations of what they do. This concept in requirements analysis can be applied by carefully studying the users' communication during work. The results of this can be used to design conceptual structures that fit into the language of users. From the study of language and how it relates to work situations using semiotic theory, requirements analysis emerges from discerning patterns of behaviour by organisational actors in the ways they behave in their work situation. The semantic analysis technique, which is the subject of section 3.5, has been advanced based on these concepts of the application of semiotic theory in requirements analysis. This section also introduces the semantic agent-based modelling formalism and its graphical representation - ontology chart - with the aid of a comprehensive example. Synthesis of both theories in one theoretical framework suggests that semantic analysis technique accords well with features of the unfreezing and refreezing stages of planned change model. Finally section 3.6 will conclude this chapter which lays down the

theoretical foundation for a new perspective on evolutionary development.

3.1 Introduction

The enhanced role of telematics - the fusion of computing and telecommunications - within the competitive strategies of firms heralds the arrival of the location-independent organisation, in which location will no longer determine planning, control, reporting, function and hierarchy, making the firm's information systems resources the real definer of organisational boundaries. The driving force behind such organisational developments appears to be the rapidly changing nature of the environment in which firms are operating, pushing organisations towards new forms of collaboration, both within and beyond the boundary of the firm. The business environment will therefore evolve from relative stability to continuous change, requiring consequent changes in business style and focus. The significance of information systems in this process of reorganisation lies in its increasing contribution to making viable and effective such collaboration between geographically dispersed participants. The implications of such reorganisation in information systems development are profound. Continuously changing businesses will have very different information needs from those of stable businesses. These needs will involve information systems in new kinds of activity in support of the business. These changes may in turn affect the way the system is used and therefore have implication for systems design. To improve the development of information systems where functions and information are increasingly distributed, we need to study new aspects of dynamic information systems development.

The central theme of this chapter constitutes a proposal for a new theoretical framework for information systems design and development in changing business environment. Chapter 2 provided some insights into the problems of the evolutionary development approach. It argued the shortcomings of the approach as:

• The risk of *ad-hocracy* which is always to the fore when using the prototyping approach, Therefore there is a need for planning the whole process of development

• The lack of a conceptual model as a frame of reference for analysis and design during the development process

This research proposes, in response to the two shortcomings noted above, a new theoretical framework based on both planned organisational change theory and semiotic theory. Adopting a contextualist framework, this research proposes semiotic theory for exploring the content of change and for continuously respecifying the information needs in relation to the context of evolutionary development. At the same time, the process of applying the content of change to that context will be managed by using a planned organisational change model. The aim is to propose a new perspective to evolutionary prototyping for dynamic information systems development.

3.2 Organisational change theories

A major deterrent to successful evolutionary system development is change in requirements and the problem of managing change. However, as discussed in the chapter 2, the more serious dimension of this problem is essentially behavioral in nature. This is because the introduction of any information system causes change in the organisation; i.e. to individuals, to responsibilities, to the socio-political structure. The purpose of the rest of this section and the next is to focus on the relationship between organisational change and evolutionary information systems development in dynamic environments. Towards this end, organisational change theories serve as the basis for a theoretical framework for analysis and management of this relationship.

Information systems have attained a level of complexity that can transcend departmental boundaries, allow communication between geographically dispersed individuals, change roles and responsibilities and even shift the power structure (Krovi, 1993). Developing information systems creates change conditions for organisations which must move from one state to another. Any new state of organisational behaviour may require more changes in the information systems itself. Analyzing the relationship between the information system development process and change induced in the environment can help to show what must be done to achieve the transition. Indeed, in some cases the

justification for introducing the information systems relies on social change taking place (Szlichcinski, 1983). It is predictable that major social and behavioral changes can stem from the introduction of new information systems and software systems. Today, software systems play a more pervasive role in the user's work. In these circumstances substantial changes in behaviour patterns occur quite quickly, well within the lifetime of an installation, as users adapt to a system and learn to exploit its capabilities. Change in business environment affects the development of information systems. It follows that development of an information system, itself, will induce changes in the workplace. Therefore, any attempt toward successful development of evolutionary systems needs to be planned and managed in order to support both modes of change satisfactorily. A carefully planned change approach is required for development of each prototype before it is implemented, installed and examined by users. A search for relevant support theories leads us to study organisational change theories and to start with, we begin with a theory of change and persistence.

3.2.1 The theoretical perspective of change and persistence

Paul Waltzlawick *et. al.* (1974) have proposed a theory of change and persistence using two abstract and general theories, the theory of Groups and the theory of Logical Types, drawn from the field of mathematical logic. As they mentioned, even if the use of these theories is far from satisfying mathematical rigour, they should be taken as an attempt at exemplification through analogy which will help and clarify the subject.

According to the theory of Groups, a group is composed of *members* which are all alike in one common characteristic, while their actual nature is otherwise irrelevant for the purpose of the theory. The grouping of things (in the widest sense) can be a collection of numbers, objects, concepts, events, or whatever else one wants to draw together in such a group, as long as they have that common denominator and as long as the outcome of any combination of two or more members is itself a member of the group. The ordering of the world into (complexity intersecting and overlapping) groups composed of members which all share an important element in common gives structure to them. While it is obvious that no two things will ever be exactly the alike, this ordering establishes *invariance* in the above-mentioned sense, namely that a

combination of any group members is again itself a member of the group. Thus this property may allow changes within the group, but makes it impossible for any member or combination of members to place themselves outside the group. The basic concepts of this theory provide a framework for thinking about the peculiar interdependence between persistence and change, while it apparently cannot provide a model for those types of change which transcend a given frame of reference. This is the idea that Waltzlawick *et. al.* (1974) have turned into the theory of Logical Types. This theory, too, begins with the concept of a collection of things which are united by a specific characteristic common to all of them. As in Group theory, the components of the totality are called *members*, while the totality itself is called *class* rather than group. According to this theory whatever involves *all* of a collection must not be one of the collection. This theory provides a paramount distinction between member and class and the fact that a class cannot be a member of itself. While we are constantly faced with the hierarchies of logical levels, the dangers of level confusions and their puzzling consequences are ubiquitous. The phenomena of change are no exception. For example as Bateson points out (1972), the simplest and most familiar form of change is motion, namely the change of position. But motion can itself be subject to change,i.e., to acceleration or deceleration, and this is a change of change or *metachange* of position. One level higher than the first one. It can be seen that in order to change from position to motion, a step out of the theoretical framework of position is necessary.

Pual Watzlawick *et. al.* (1974) found those two theories complementary and they can equip us with a conceptual framework useful in examining concrete, practical examples of change. According to this framework, Group theory gives us a framework for thinking about the kind of change that can occur within a system that itself stays invariant; but the theory of Logical Types is not concerned with what goes on inside a class, i.e., between its members, but gives us a frame for considering the relationship between member and class and the peculiar metamorphism which is in the nature of shifts from one logical level to the next level up. As they mentioned, if we accept this basic distinction between the two theories, it follows that there are two different types of change:

- One that occurs within a given system which itself remains unchanged.

They refer to this kind of change as *first-order change*.

• And one whose occurrence changes the system itself. This kind of change is referred to as *second-order change*. This is thus change of change or metachange.

Therefore, changes in the body of rules or norms governing the structure or internal order of groups are the subject of second-order change level, where that groups are invariant only on the first-order change level i.e., on the level of change from one member to another.

Researchers from different fields have also made a distinction between two types of change, specifically in the area of organisational studies and management. Although, these definitions complement each other, there are differences. Most of these studies have been classified in a sub-area of management known as organisational development. The fields of psychology, sociology, management, and organisational behaviour have contributed to the formation of an interdisciplinary approach to handling organisational adjustment to change. Today organisational development is an evolving field which emphasizes the importance of the human dimension in the act of introducing and adapting to change (Desanctis & Courtney, 1983). From that school of thought, Greiner (1972) has defined evolutionary and revolutionary changes. He named the evolutionary change as the adjustments necessary for maintaining growth under the same pattern of management in the organisations, while revolutionary changes are the serious upheavals and abandonment of past management practices which involves finding a new set of organisational practices.

The other definition for levels of change in organisational development studies is the classification of change to Alpha and Gamma changes introduced by Golembiewski *et. al.* (1976). In their words, Alpha change involves a variation in the level of some existential state and Gamma change involves a redefinition or conceptualization. Other definitions are similar: single loop learning and double loop learning (Argyris & Schon, 1986), equilibrium systems and far from equilibrium systems (Goldstein, 1988),

incremental change and transformational change (Kindler, 1979), etc. (Krovi, 1993).

3.2.2 Third order change and evolutionary information systems

The two types of change identified by different researchers were introduced, as they represent two extreme ends of a spectrum; i.e. one end deals with incremental change, the other implies a radical change (Krovi, 1993). Some researchers have attempted to define a new type of change as third level change. Golembiewski *et. al.* (1976) defined a third level change as Beta change which involves a variation in the level of some existential state, complicated by the fact that some intervals of the measurement continuum associated with a constant domain have been recalibrated. However, when Alpha and Gamma changes indicate the consequences of using the system, Beta change refers more to the *process* of attaining a particular level of change. Bartunek and Moch (1987) also introduced a new order of change to the body of existing literature about organisational change. They called it third order change which requires the consultant to play of more a teaching role, training the client system to distinguish among schemata and develop and implement alternatives. In this type of change, the instrument used to measure the change has itself undergone change (Krovi, 1993). This level of change does not fit into the spectrum between first and second order change. It is a circumstance where a continuous feedback loop operates between the change agent and the environment to achieving a particular change. Managing these feedback interactions toward a desired situation is the main target of organisational development studies.

The three levels of change can be compared through their potential effects on the environment. In this sense introducing information systems, or specifically computer-based information systems, may cause different levels of change in the organisations; hence we can study evolutionary information systems according to their impact on the environment and the level of change they induce in different activities of the organisations. This study leads us to classify evolutionary information systems as third order information systems corresponding to third order change.

First order information systems development provides an incremental change and might

create a minor change in organisational processes. The main objective of these systems can be a change in the level of fine-tuning, fixing problems, making adjustments, modifying procedures. The driving force behind developing these systems is improving efficiency or even on an *ad hoc* request. This type of information systems is not intended to modify any critical success factor of the organisations and can have low potential effect on the individuals and also low direct impact on the environment. The system has no lasting effect on its environment and the environment also has no effect on the system. In this type of system a fully formulated, stable system can be engineered to precise requirements. The traditional, highly structured and deterministic approach to development coincide remarkably well with this situation. It is like interaction within the members of a group.

In contrast, the *second order information systems development* might revolutionise the process used to conduct the substantive business of the organisations. The main objective of this system is fundamental and large-scale change, a transformation, a refocus or reorientation. Computerised banking system and the introduction of Automatic Teller Machines defined new success factors for banks and forever changed operations in the banking industry. Here the information system has the dominant role in inducing change and the environment follows the changes to achieving the next logical type. This type of information system will modify the critical success factors of organisations and provide high impact on the environment and will affect many people. It is the situation of inducing change deliberately through the development of proactive information systems. But sometimes second level changes are caused mainly by factors which are external to the information system and its environment. In these circumstances, such as a major reduction in market share, some sort of major crisis and strategic change will be the drivers behind developing the information system in order to facilitate the management of the change, or as a reaction to the external factors. We can classify such changes also as second order information systems, not because of their potential for inducing the second level of change to the organisations, but due to the fact that they also become part of the causes of inducing second order changes to the firm. They don't have the total role of the change agent, but they are some part of it. The external factors in the environment provide a continuous positive feedback for

major changes to which the information system also need to adjust. From the information system developer's point of view these factors play the role of second order change agent for both the information system and the organisation. Both the information system and the organisation are in the reactive situation. The external environment has the dominating role in the relationship between the formal system development and its associated informal system. Strategic information systems are the typical examples of this type of second order systems. Second order information systems are indeed the focus of a subject known as business process re-engineering (BPR). They are carriers of sweeping change.

Third order information systems development are the systems where the magnitude of their ability to change is greater than first order information system, yet they neither affect the critical success factors nor are revolutionary in nature. The need for this type of systems is also mainly driven by operational efficiency considerations. But the initial requirements are only relevant for developing the first version of system and during its operational life the system will need to respecified until some stability occurs in the relationship between information system and its environment. Third order information systems are evolutionary in nature. The implementation of this type of evolving systems are the most important area of using evolutionary approach through prototyping. As an example, the introduction of electronic mail systems had an organisation wide impact and affected some employees but did not change the whole business. Third order information systems are evolving systems with a high interaction between the system and its environment. While the system involves a change in the level of some current state, the whole organisational evaluation and expectations themselves undergo change. The organisational use of the information system evolves through time while the information system specification and its impacts also evolve. For example, the organisational use of electronic mail systems has evolved to support interfirm electronic data exchange and the transaction and coordination costs have been reduced immensely through this type of connectivity. This in turn, alters the organisational boundaries and provides new arenas of demand for technology. This type of information systems will need to be respecified continuously while the evaluation of their associated impact is recalibrated.

Management information systems (MIS) are also another typical example of the third order information systems. The degree of difficulty associated with realizing the intended benefits of these systems tends to be directly related to their complexity, which are comprised of technical and organisational aspects. Generally, technical complexity is related to the size of system, volume of data flow and system interfaces required. But organisational issues refer to the dependence of the organisation upon the system, its integration with other organisational systems, its impact upon the duties and responsibilities of organisational members, and its impact upon socio-political structures (Zmud & Cox, 1979). These technical and organisational complexities are rooted in the interactions among planned information systems, among existing formal and informal systems and external environment, while understanding the system requirements itself is the subject of evolving change through the use of the system.

3.2.3 Evolutionary information systems implementation

Using the change approach for information system classification and concentrating on third order information systems as an emerging demand in information system development leads us to focusing more on the implementation *process* itself. The discussion of information system development as inducing change to the firms can be expanded to the process of change itself. This in turn will provide a set of guidelines in order to manage and improve this process.

While it is practically a cliche to state that change in organisations today is a way of life (Goodstein & Bruke, 1991), several authors conclude that introducing a computer-based information system into an organisation results in profound changes to the social as well as technical fabric of the organisation (Boland, 1978; Bostrom & Heinen, 1977; Ginzberg, 1978b; Zmud & Cox, 1979). Ginzberg (1978a) holds the view that it is the intention behind many information systems projects to change the role behaviours of organisational members. Boland (1978) has identified the change approach to information system development as a protocol of implementation in comparison with a traditional approach. While the traditional approaches place more stress on the prescriptive implementation stages with a passive role for user involvement, the change approach shows a greater concern for the beginning and ending stages of the change

process, i.e., initiation, conversion, and evaluation (Zmud & Cox, 1979) with the active participation of users. That is, while earlier approaches to implementation focused on measuring and classifying, the change process approach focuses on managing the process. It means user and system analyst have a role of joint change agents in order to discover an appropriate change level through mutual teaching and criticism.

In order for participants to be able or willing to contribute to an implementation process, they must understand the semantics of ongoing change and how the project will affect their organisational role during and after implementation. Without such knowledge, ignorance and uncertainty will lead to resistance to involvement and eventual disassociation from the information system project (Dickson & Powers, 1973; Dickson & Simmons, 1970). The system analyst cannot truly assume responsibility for another person's behaviour. Responsibility for internalizing required behaviour patterns, therefore, must lie with the user (Bostrom, 1989; Bostrom & Heinen, 1977). Any successful change needs an environment in which change will be accepted through the active involvement of affected organisational members. While many information systems implementation may radically alter the duties and responsibilities of organisational members, they require conceptual training that provides an overview of the future system as well as specific instruction pertaining to each individual's personal relationship with the system. These individuals must be informed of their roles in the conversion effort as well as of what to expect during conversion. The resulting impact of third order information systems implementation affects both formal and informal relationships with, and attitudes toward, the organisation (Zmud & Cox, 1979).

The organisational environment is affected through development of evolutionary information systems as informal systems are gradually formalized and organisational functions are integrated. If the change process cannot involve responsible agents in establishing a mutual trust among all participants about their future role so that a free exchange of ideas become possible, organisational members may not fully understand why change is occurring - often leading to misconceptions, misuse, and mismanagement of the future system. These are issues which highlight the importance of management of the implementation process in evolutionary development. This leads us to focus

more on change process theories which are the subject of the next section.

3.3 Change process theories

A fundamental taxonomy of change theories includes process theories and theories about possible levels of change. The former deal with how change can be attained and the latter with what can be achieved (Krovi, 1993). The previous section looked at the theory considering different levels of change. This section focuses on process theories in order to introduce the *planned organisational change* approach for implementation management of the third order information systems development.

3.3.1 Change Process theories and implementation of evolutionary information systems

Implementation is inherently a dynamic phenomenon; the state of a given factor can change or be changed in the course of the implementation process, and so no snapshot view can possibly represent the entire process. We explicitly view implementation as a *process*, and examine information systems implementation efforts in the context of dynamic process models.

Ginzberg (1978b) proposed two theoretical bases as the source for dynamic process models; planned organisational change and the adoption of innovation:

- **Planned organisational change**

  There are two seminal models of implementing planned organisational change related to the information system development process which were proposed by Kurt Lewin (1952)/Edgar H. Schein (1961, 1972) and Kolb and Frohman (1970). The Lewin/Schein model suggests that any change effort can be viewed as including three distinct phases: *Unfreezing, Moving, Refreezing*. Each phase is concerned with changes in the balanced of forces existing in the organisation, and the degree to which they foster change or resistance to it (Ginzberg, 1978b). Unfreezing means creating the need for change, Moving is choosing a particular course of action and implementing

it. Refreezing entails bringing back the organisation to stability. The Kolb and Frohman model of the consulting process (Kolb & Frohman, 1970) considers the interaction between the client and the consultant and sees the implementation process as consisting of seven stages (Krovi, 1993). This seven stage elaboration of Lewin/Schein theory (Lewin, 1952; Schein, 1961, 1972) was proposed as a normative model of the implementation process. Ginzberg (1978b) has compared the two models by matching the corresponding stages of implementation.

- **The innovation process**

The innovation process approach consists of extended models of innovation in organisation which are evolved from the early work in rural sociology (Wolek, 1975). These models delineate a sequence of steps which are followed in the process of adopting an innovation. In general, however, they begin with the recognition that a problem or opportunity exists, and then move on to developing an awareness of a potential solution, and eventually reach a trial, then sustained, usage phase for the innovation (Ginzberg, 1978b).

Although all of the above approaches were derived from separate line of inquiry, the characterization of the process of change is quite similar in all of them. This type of *process-oriented view* to implementing planned change leads us to consider the entire evolutionary information systems implementation process - from initial inquiry and change planning to installing and evaluating the changed state - rather than only the action stage, which is sometimes viewed as synonymous with implementation. Many of the problems which manifest themselves late in an information systems development projects actually have their roots in an earlier stage (Ginzberg, 1978b).

3.3.2 Lewin's change process model

According to an open systems view, organisations - like living creatures - tend to be homeostatic, or continuously working to maintain a steady state. This helps us understand why organisations require external impetus to initiate change and, indeed,

why that change will be resisted even when it is necessary (Goodstein & Bruke, 1991).

Kurt Lewin's (1952) three-phased model of change - unfreeze, move (or change) and refreeze - suggests that the first step of any change process is to *unfreeze* the present pattern of behaviour as a way of managing resistance to change (Goodstein & Bruke, 1991). Whatever the level of change involved, any information systems intervention is intended to make organisational members address the need for that level of change, to heighten their awareness of their own behavioral patterns, and to make them more open to the change process. The first phase of change, entails the disconfirmation of existing, stable behaviour patterns, establishing a "felt need" for change (Ginzberg, 1978b).

The second step, *movement* involves making the actual change that will move the organisation to another level of response (Goodstein & Bruke, 1991). This second step is the action phase of the change effort. This requires the presentation of information necessary for change and the learning of new attitudes and behaviours which are necessary parts of the change (Ginzberg, 1978b).

The final stage of the change process, *refreezing*, involves stabilizing or institutionalizing the change by establishing systems that make these behavioral patterns "relatively secure against change", as Lewin (1952) put it. During the refreezing stage, the organisation may also ensure that the new behaviours have become the operating norms at work (Goodstein & Bruke, 1991). Refreezing entails the integration of new attitudes and behaviours into persisting patterns and relationships (Ginzberg, 1978b).

It is important to mention that the whole sequence of unfreezing, moving and refreezing must be seen as an iterative process, and will likely be repeated more than once in any sizeable change effort, such as the implementation of a large or complex information system (Ginzberg, 1978b). Zand and Sorensen (1975), in their research on operationalizing the Lewin's (1952) theory, suggest that unfreezing is conducive to moving, and moving would be conducive to refreezing, and all three stages are positively correlated, as resistance to them will also be positively correlated. The result

of their research also showed that there was a tendency for poor performance at one stage to be followed by poor performance at the later stages.

Implementation of information systems in organisational context means moving from a known present state to a desired future state. Therefore organisations must recognize that the intervening transition state requires careful management, especially when organisational change is large and complex (Goodstein & Bruke, 1991). To facilitate this transition state, often characterized by temporarily lowered effectiveness and disorganisation, system developers must decide at the outset of a project when they are going to be a change agent and interventionist. To intervene is to enter into an ongoing work system for the purpose of improving its function (Bostrom & Heinen, 1977). Being a change agent requires that the system designers really come to understand the underlying business activities and make sure that the project is going substantially to set the desired norms and rules within the organisational context. It necessitates users involvement throughout the entire project, specifically regarding their contribution to the job of implementation. It needs mutual understanding between users and system developers.

The semantic analysis technique, derived from semiotic theory, can be useful in supporting design understanding during the unfreezing and refreezing stages of change. While requirements prototyping is mainly concerned with the change (movement) stage of the planned change approach, semantic analysis technique provides an environment in which change can be accepted through the conceptual training of those organisational members affected. Open systems thinking, a planned model of managing change, and the theory of multiple levels of change can be seen as elements of a new perspective in information systems process management and implementation. The change effort heavily relied on the understanding about the nature of organisations and changing them. This leads to the suggestion of semiotic theory as the subject of the next section.

3.4 Semiotic theory

Semiotics, "the science of the life of signs within society" as Saussure (1966) defined

it, can provide a theoretical framework for analyzing and understanding information requirements. The common dominator of different kind of signs is that they stand for something else than themselves. Our world is full of objects that are used as signs: words, pictures, facial expressions, body postures, films, traffic lights, uniforms, etc. When a sign occurs, two entities occur: an *expression* and a *content.* These two planes of signs are sometimes called the signifier and the signified.

To give a simple example, tossing a coin on the table may not be a sign. If we just want to get rid of the coin, then only one act occurs: tossing the coin. However, this act can be used symbolically to settle a dispute. Then one side of the coin acquires the content "I win" and the other side "You win", and now two acts occur simultaneously: the coin is still tossed, but now accompanied by the element "You win". This double entity is called a sign, or more precisely a sign relation, which is a relation between a content (the signified) and an expression (the signifier). Both expression and the content must be present in a sign (Andersen, 1991a).

Semiotics must necessarily view computer systems as sign-vehicles whose main function is to be perceived and interpreted by some group of people (Andersen, 1991b). In this context information systems are sign systems, while computer systems are symbolic tools. Semiosis, the process of sign formation and interpretation, distinguishes between two sets of characteristics of information systems as sign systems. The first set comprises the surface structure characteristics of information systems. These characteristics manifest the representation and form of signs. The second set comprises the deep structure characteristics of information systems. These characteristics manifest the meaning of signs and the agent who uses these signs to symbolise an intention. Signs and their meanings are inextricably linked. More specifically, signs and their meaning-in-use may be understood in terms of four levels; corresponding to the four major branches of semiotics (Stamper, 1987): pragmatics and semantics (deep structure characteristics of signs), syntactics and empirics (surface structure characteristics of signs). The entire structure presupposes that responsible agents, which might be individuals, groups, or larger organisations, have commitments, expectations and relations within social frameworks. These reflect the ability of actors with thoughts to

have an effect upon the world (Liebenau & Backhouse, 1990). These four levels of emergent properties of signs, may be briefly summarised as follows, as they relate to the information systems context.

At the most basic level, signs may be described in terms of their *empirics*, i.e. their physical characteristics, including that of the medium used in their communication. At that level, attention focuses upon the very limited set of questions about the repeated use of sign in statistical terms (Stamper, 1987) what has, regrettably, become known as information theory. In relation to information systems, empirics is essentially concerned with signs as signals and codes. We can use empirics to analyze signalling, computing and communication hardware requirements and the actual signals generated by software instructions at the machine level.

The second level, i.e. *syntactics*, concerns the formal rules which govern the use of signs. By formalizing, we provide rigor to the use of language by the constraints of vocabulary, grammar, and rules which govern them (Liebenau & Backhouse, 1990). With respect to information systems, this level is concerned with data model and conceptual schema, operating system software, and programming language environments.

The third level, i.e. *semantics*, deals with the issue of meaning, the relationships between signs and what they purport to represent (Stamper, 1987), i.e. their referent. As Liebenau and Backhouse (1990) have argued, what is crucially different here from commonly held notions of meaning is the rejection of the idea of an intrinsic meaning to a sign, and its replacement by a model which relies upon two agents or groups interacting in a complex exchange whose effectiveness is tested in the actual behaviour of the parties involved. With respect to information systems, the semantic level of signs involves agreeing upon boundaries, identifying individuals, establishing and maintaining classifications (Stamper, 1987).

The fourth level, i.e. *Pragmatics*, is concerned with the context of activity, and those characteristics of people, organisations and acts of communication which affect

information (Liebenau and Backhouse, 1990). With respect to information systems, this level is concerned with cultural context and norms and with the intentional behaviour of organisational agents in terms of which the meaning of signs-in-use may be specified.

Both users and designers interpret the information requirements in an effort to implement an information system, and the clashes between these two kinds of interpretations and also among users themselves are an interesting topic. In fact, the main purpose of applying semiotic theory into information systems development is to contribute to a framework for connecting systems interpretation with system design (Andersen, 1990). Semiotic theory focuses on the users' own interpretations of what they do rather than on methods and tools that pretend to give an objective account of the work process. The users' own representations can be investigated by carefully studying their communications during work. The results of this can be used to design conceptual structures that fit into the language of its users and to compose powerful symbolisms. From the study of language and how it relates to work situations, requirements analysis emerges from discerning patterns of behaviour by organisational actors in the ways they behave in their work situation and talk about it. In this account, language is seen as a social phenomenon and is described according to the functions people use it for in real life. This makes semiotic theory suitable for describing communication in work situations. In this respect, the semantic aspect of language has the highest priority in requirements analysis.

The semantic analysis technique (Stamper, Backhouse, & Althaus, 1989), which is the subject of next subsection, has been developed from the application of semiotic theory in requirements analysis. This analysis technique respects actual language usage as the basis of analysis.

Although in semantic analysis, the emphasis is on semantics, a set of fairly explicit rules in a form of a modelling formalism has also been developed for relating meanings to observable expressions and language to social structures. This means that semiotic theory can be used in practical textual and communication act analyses to lift the notion

of conceptual modelling from the syntactic to the semantic level.

3.5 Semantic analysis

It has been suggested that information system be considered as a set of, essentially, arbitrary signs whose emergent properties, i.e. syntactics, semantics and pragmatics, are intersubjectively negotiated between intentional organisational agents and, as such, inseparable from the forms of social life which they sustain and in which they are generated.

Semantic analysis is a technique for specifying the information requirements of an organisation or business. By defining the business in terms of what actions and behaviours are required of the persons that comprise it, a robust yet redefined specification is created which can be used for understanding the organisation and developing information systems. The technique deals directly with the vexing problem of differences in meaning and can therefore be used to support the process of interpreting.

The aim of employing semiotic theory in requirements analysis is to focus on responsible agents who uses the sign to symbolise an intention. In order to signify and express their intentions, the responsible agents (parties) involved must have recourse to signs - explicit mechanisms - which permit the communication of intentions to take place. It is at this point that we encounter the problem of meaning. The agent who uses the sign to symbolise an intention must rely upon some social norms in his working environment to interpret the sign, whether natural language, a gesture or some symbolic pattern of behaviour (Backhouse, 1990).

Working in a business environment and solving the practical problems of day-to-day business, responsible agents acquire a set of norms of perception, evaluation, cognition and behaviour. Language does not enter into all these social norms but into a high proportion of them, and amongst them are the language norms to which we shall refer in requirements determination. Revealing the language norms should be a primary

concern of any requirements analysis effort. The meaning of words depends upon the contexts established by the actions to be performed. By placing the notion of semantics in this context, we are able to focus our attention on the correct operational links between words used to represent requirements and the things they refer to in the world of actions. In this respect, semiotic theory supports a **semantic analysis** technique to accommodate the varying operational meanings of requirements which correspond to the linguistic norms of different subsets of the user population, who are trying to solve different problems but are using the same words. In this analysis, we are looking at the ways in which the requirements are linked operationally to the real world. Responsible agents and their actions are the key aspects of an operational semantics.

Semantic analysis is a tool which can support analysts trying to represent a multi-subjective reality and to detail the connections between the signs used in organisational communication and the behaviour to which they refer in the world of action. The analysis aims to provide a precise model of the information needs for the organisation, by expressing in a relatively formal manner what the organisation actually does. Rather than attempting to capture the data in the organisation, semantic analysis aims at representing the responsible agents and the range of their possible behaviour and actions. By specifying the underlying business tasks in this manner, the information requirements of the organisation can be addressed without prior commitment to any particular computer data model or business procedures (Backhouse, 1990).

Semantic analysis itself employs a **semantic agent-based modelling formalism** to any given problem. The formalism provides a baseline for understanding the meaning of requirement changes. It gives a conceptual model which can define the information requirements. Unlike the mechanistic techniques that emphasise fixing and automating an information system at the cost of neglecting the organisational nature of information, semantic analysis technique has an organic, fluid and reconfigureable character. It pinpoints the underlying prime tasks of organisations which are less likely to be the subject of change. Performing semantic analysis requires that the analyst reconfigure the domain under study in terms of the semantic primitives in the shape required by the formalism of semantic agent-based modelling.

At the simplest level, semantic analysis involves modelling all actions, behaviours and responsible agents which characterise a given organisation and arranging them in a sequence of existence dependency. Semantic agent-based modelling provides a dynamic schema of relatively autonomous semantic units. These semantic units are graphically represented in the form of a chart, namely the **ontology chart**. Each semantic unit in an ontology chart is typically a specific thing in reality. This formalism by incorporating a theory of meaning into the modelling grammar will make it clear how the data in a software system relate to the actions to be performed in the business.

3.5.1 Semantic agent-based modelling

Performing semantic analysis entails the application to the problem of the semantic constraints of a specification tool. Its formalism, semantic agent-based modelling, has a vocabulary and a grammar of its own represented in a graphical form. The task in semantic analysis is to take the signs used in discourse in organisation and recast them within the specification formalism of semantic agent-based modelling. We may work from textual material, observation or interviews with users. The aim is to produce a representation of the business where the terms used to describe the organisation are semantically normalised, that is, subjected to rigorous constraints that ensure no ambiguity exists. Table 3.1 provides an overview of design constructs derived from semantic agent-based modelling formalism. The suggested conceptual practice is based upon two philosophical assumptions (Stamper, Backhouse, & Althaus, 1989):

- there is no reality without an agent, and
- the agent constructs his reality through his actions.

The first assumption says that there is no objective reality. One cannot separate the information from the people that use it. Information is relative to the one that interprets this information. Different people may have different views of the world. According to second assumption, information is not some kind of mystical fluid that can be send across telecommunication lines or stored in databases. Information is a semiotic sign that affects the behaviour of the agent taking notice of it (Nauta, 1972).

Table 3.1 An overview of design constructs in semantic agent-based modelling

Affordance class	An affordance class is the basic unit of analysis in our ontological chart. A real world is made up of affordances. All social and physical environments afford certain ways of acting or behaving and each behaviour the environment affords is an affordance. These affordances are the primitives of semantic analysis. Each affordance is an instance of a class affordance. We are mainly interested in affordance classes as a matter of abstracting the problem domain.
Dependency	Two affordances are said ontologically dependent, if the existence of one of these is a necessary precondition for the existence of the other. The affordance that is a precondition will be called antecedent and the affordance depending upon its prior existence will be called the dependent.
Period of existence	The antecedent must exist during the whole period of existence of the dependent, not just during the start or finish of its existence.
Agent	An agent is a special type of affordance who is holding responsibility in the world of actions. Agents are legally, or socially in organisational terms responsible for any particular portion of real world activities. According to the philosophy of semantic agent-based modelling there is no reality without an agent, and the agent constructs his reality through his actions.
Pseudo-agent	Agents are in principle human: a person or a legal individual (legal person) like a company who can hold responsibility in the legal or social sense. However, formalism tends to veil responsibility by making it possible to shift the onus onto a machine. Formalism simply embodies the value of the formalizer. In this respect machine can be seen as pseudo-agent which governed by rules developed by a responsible rule-giver (agent).
Whole-part	A whole-part relationship is also an ontological relationship: a part cannot exist without the prior existence of a whole.
Generic-specific	A generic-specific relationship between different affordances comes from the assumption of hierarchical groupings and inheritance of properties. Sometimes, a certain way of behaviour is open to more than one of the recognised agents.
Universals-particulars	Affordances may be universal or particular. The universal affordance connotes some invariant patterns of behaviour in a domain without referring to any specific occurrences, whether actual or hypothetical. All responsible agents (and pseudo-agents) are particulars and we can refer to them particularly.
Individuation	Particulars which can be measured and taken apart from a set, can be individuated. When recording facts about particular individuals, we use the notion of individuality which presupposes that we are able to recognise the uniqueness of each discrete individual.
Identification	Particulars are called identifiable, if there are a general form of measurement, or determination. This measuring system can be very abstract which can be used for referring to any particular *particular.*
Sign type	Sign type is a semantic affordance which contains a recognisable pattern or shape regardless of the physical medium in which it may be realised.

The meaning of information can provide ontologically expressive semantic agent-based modelling constructs through using a specific grammar. This grammar which is based on the design constructs represented in table 3.1 consists of a basic unit of script namely **affordance** (Gibson, 1977). For any single agent an invariant pattern of behaviour is referred to as an affordance. The environment **affords** the agent this behaviour. All social and physical environments afford certain ways of acting and behaving, and these affordances are the primitives of semantic modelling. The affordances have to be arranged in a manner whereby the dependency of one affordance upon the prior existence of others is detailed. This ontological dependency gives rise to the name for one representation of the resulting semantic schema: **the ontology chart**. Ontology charts depict the range of affordances possible in any given domain.

Ontology charting based on the semantic agent-based modelling formalism will be presented in subsection 3.5.3. But before that it is important to describe the key issues in conducting a semantic analysis and its important characteristics which make the technique viable in specifying user requirements in business change environment. This is presented in some detail in the next subsection.

3.5.2 Key issues in semantic analysis

The model of information systems that is proposed in semantic agent-based modelling is one that demands attention to semantics of the signs used by the participants. The followings are the key issues in conducting semantic analysis and its formalism (Backhouse, 1990):

- **not inventing a new vocabulary**

  In using the terms that are normally associated with the work of an organisation, the analyst maintains the established and natural signification of the particular vocabulary that has developed in that context. By introducing alien terms, often dictated by the limitations imposed by operating systems upon the length of filenames, the analyst increases the difficulty of communication, and the likelihood of ambiguity.

• information as a social process

Information is the culmination of the process of communication, where the participants have successfully achieved the appropriate interpretation of the signs employed in any discourse. Data are signs in a message or recording system whose referents are found in the actual organisation. Without a context and an agent to interpret, information cannot result from data. We are interested primarily in the content of the messages and records, i.e. in the substantive concerns of the organisation and how to interpret them.

• subjective context

One model of an organisation yields only a view of that reality; the process of semantic agent-based modelling may reveal several, possibly conflicting, views. It may be difficult to agree upon on an objective picture of an organisation and its activities. Semantic analysis seeks to identify who is responsible for determining when any of the elements modelled in an schema actually come into, or go out of, existence. In this way we are placing responsibility to maintain semantic integrity within the domain under study.

• responsibility instead of truth

For each affordance in the model of the organisation we will need to know who is responsible for the realisation of it, who determines the existence. In this sense we want to pin back the formal part of our model onto those who have to accept the responsibilities which arise when affordances become realisations, when the model gives way to realised instances. Rather than truth as the underlying, but elusive, notion governing the semantic integrity of what we define, we have instead that of responsibility.

• multiple definitions: negotiation

Close inspection of any large organisation will usually reveal that different meanings are being attached to terms in common use throughout. One department will have a different perspective from another on what

constitutes the precise definition of a given term. When this occurs the negotiation of meaning can take place. Triggering this type of negotiation and resolving the ambiguities through finding responsible agent for realisation of any actions and behaviour is crucial task of a semantic analyst.

• any model has meta-level assumptions

Any analytical technique rests upon the underlying assumptions from which it arises. These assumptions, sometimes referred to as the meta-model, are automatically incorporated into any particular models which we construct using that technique. The key question here is to what extent do the underlying assumptions permit a penetrating analysis which can aid managers in understanding the core nature of their organisations and in defining the information requirements for any computer data systems they need.

• substance not procedure

Organisational conventions and business procedures tend to change rapidly, along with continual development in technology. Any business model which incorporates procedure into the model will constantly have to be retouched to cater for new contingencies (hence the software maintenance backlog). Our objective must be to go beyond the procedural level of dataflows and functional analysis, and reach down to, and represent the substantive level of the business operation. In this sense the model of information requirements remains independent not only of any physical configuration constraints but also of any data constraints which derive from current mode of business operations.

• reusable analysis

Where a piece of organisational behaviour has been proven over time to be effective, it is not surprising that we can expect to reuse it in similar environments. This can provide a powerful mechanism for tracing back

each piece of software to the semantics of its underlying information requirement within the application domain which will enable the achievement of yet higher levels of requirements traceability and software reuse.

- **time**

One universal facet of social and organisational life which is repeatedly either ignored or mishandled by specification techniques is that of time. Occasionally time is introduced as entity in its own right, as an attribute/field, along with all other elements that specify the organisational activity. Rarely is time handled at the meta-model level: modelling techniques in general cannot represent the dynamics of the schema itself, even though changes in the data serve to reflect changes in the real world of the organisation. What is needed is an *a priori* assumption about existence: a meta-model which assumes that all phenomena experienced by knowing agents have a lifespan which is determinable by these agents.

3.5.3 Ontology charting

As said in subsection 3.5.1, semantic analysis technique focuses on linguistic categories of signs and signifiers. The purpose of this analysis technique applied by the semantic agent-based modelling formalism is to make an ontology chart of the domain focusing on semantics and ontological dependencies. Ontology charts are a way of representing a domain of behaviour, and are usually used to show the results from semantic analysis. One result of a semantic analysis is that an organisation is represented as a collection of sets of behaviour, shown graphically. The basic element of each set is a node set in a network of behaviours. Each node represents a pattern of behaviour which persists in the organisation, but the position of each pattern is important - what is to left on the chart is behaviour that must be realised before what is to the right may be accomplished.

We are imposing semantic constraints which specify the existence of requisites for any given behaviour upon the representation in a way that closely resembles the logic of the

world of action. In order to achieve or realize certain behaviours, other behaviours must be realized first. This concept of existence constraints is a concept of ontology. To use the concept schematically gives us ontology charts (Backhouse, 1990).

The ontology chart is based on a very particular kind of relationship that we can observe in the world around us: the ontological dependency relationship. The cornerstone of this modelling relation is that two things are ontologically dependent, if the existence of one of these is a necessary precondition for the existence of the other. The thing that is a precondition is called the *antecedents* and the thing depending on it the *dependent.* It is important to note the antecedent must exist during the whole period of existence. Therefore, ontological dependency is not the same as causality (Thonissen, 1990).

For example each country has different regions. Region is an affordance in this association. The sematic agent-based notation for an ontological association is a line between affordances. This type of *unary association* describes an ontological link between country and region. Country is the antecedent of region, because there is no region of a country without a country. The country *affords* this behaviour of having different regions. As shown in figure 3.1, it is necessary to arrange the affordances to read from left-to-right according to their ontological dependencies.

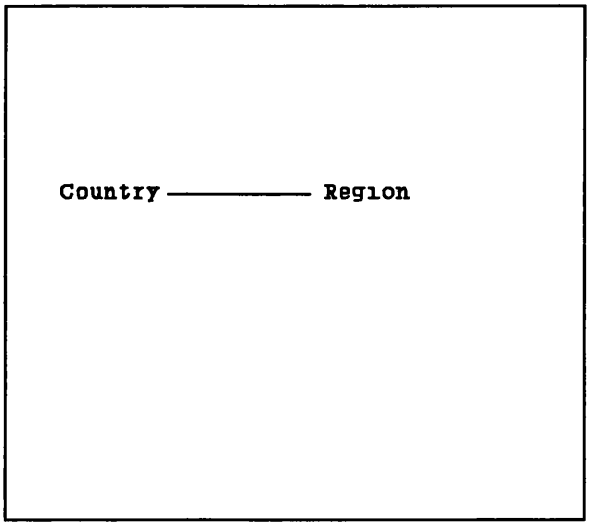


Figure 3.1 Unary association in ontology chart

Ontological dependencies can only be either unary or binary association. As another example, a citizenship can only exist during the period that both country and a person co-exist. This type of ontological relationship is called *joint affordances* or *binary association.* Country and person are the antecedents of citizenship; and citizenship is

their dependent because there is no citizenship possible without a country and a person. If the country drowns or ceases to exist, the citizenship of that country necessarily stops as well (as with the establishment and dissolution of the Soviet Union and the establishment of the Ukraine, Russia and so on). And the same for the existence of a person as a citizen of citizenship binary association (see figure 3.2).

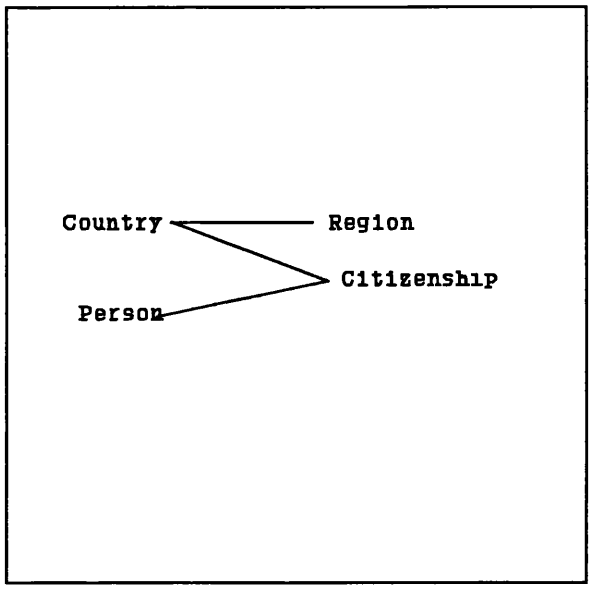


Figure 3.2 Binary association in ontology chart

It is necessary to explain that the abbreviation of **affordance** is used in this thesis instead of **affordance class**. The notion of abstraction is at the heart of definition for affordance classes. An affordance class is an implicit property of each affordance. By grouping affordances into classes, we abstract a problem. Abstraction gives semantic agent-based modelling its power and ability to generalize from few specific cases to a host of similar cases. Common semantic links are stored once per class rather than once per instance. So ontology charts are **class diagrams** for describing many possible instances of affordances. We can also have **instance diagram** to describe how a particular set of affordances ontologically link to each other. Instance diagrams are useful for documenting test cases (especially scenarios) and discussing examples which help to resolve ambiguities. A given class diagram corresponds to an infinite set of instance diagrams. Figure 3.3 shows an instance diagram of the class

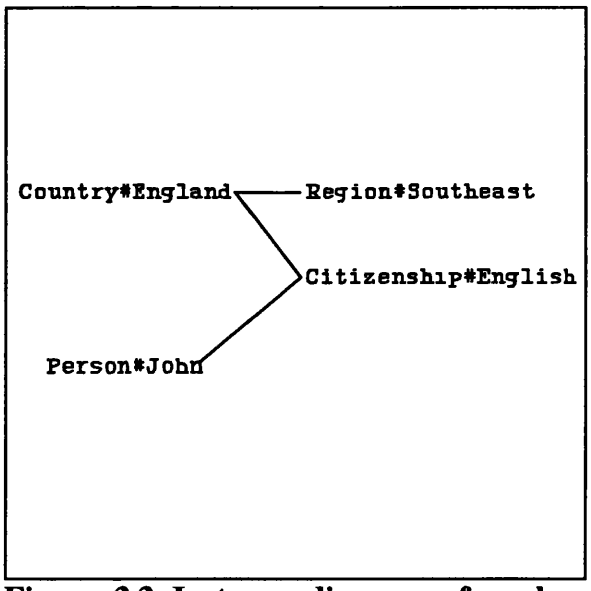


Figure 3.3 Instance diagram of a class diagram

diagrams in figures 3.1 and 3.2.

Class diagrams in the form of ontology charts describe the general case in modelling a system. Instance diagrams are used mainly to show examples to help to clarify a complex class diagram. The distinction between class diagrams and instance diagrams is in fact artificial; affordance classes and their instances can appear on the same diagram, but in general it is not useful to mix classes and instances.

In addition to affordance, there are other semantic agent-based modelling constructs which were represented shortly in table 3.1. In order to begin to illustrate semantic agent-based modelling grammar and how semantic analysis works, this subsection introduces an example drawn from the banking system.

An example of automated banking system

The following problem statement for an automated teller machine (ATM) network shown in figure 3.4 (adapted from: Rumbaugh *et. al.*, 1991, Object-oriented modelling and design, p.151) serves as an example throughout the rest of this subsection.

The aim is to represent the concepts of semantic agent-based modelling formalism. It shows the practicality and usefulness of semantic analysis and provides some guidelines in developing ontology charts as the results of that analysis. Ontology charts are useful both for abstract modelling and for designing actual systems. Ontology charts are concise, easy to understand, and work well in practice. The formalism of semantic agent-based modelling is illustrated by ontology charts to introduce the notation and clarify our explanation of concepts in semantic analysis.

Problem statement: Design the software to support a computerized banking system including both human cashiers and automatic teller machines (ATMs) to be shared by a consortium of banks. Each bank provides its own computer to maintain its own accounts and process transactions against them. Cashier stations are owned by individual banks and communicate directly with their own bank's computers. Human cashiers enter account and transaction data. Automatic teller machines communicate

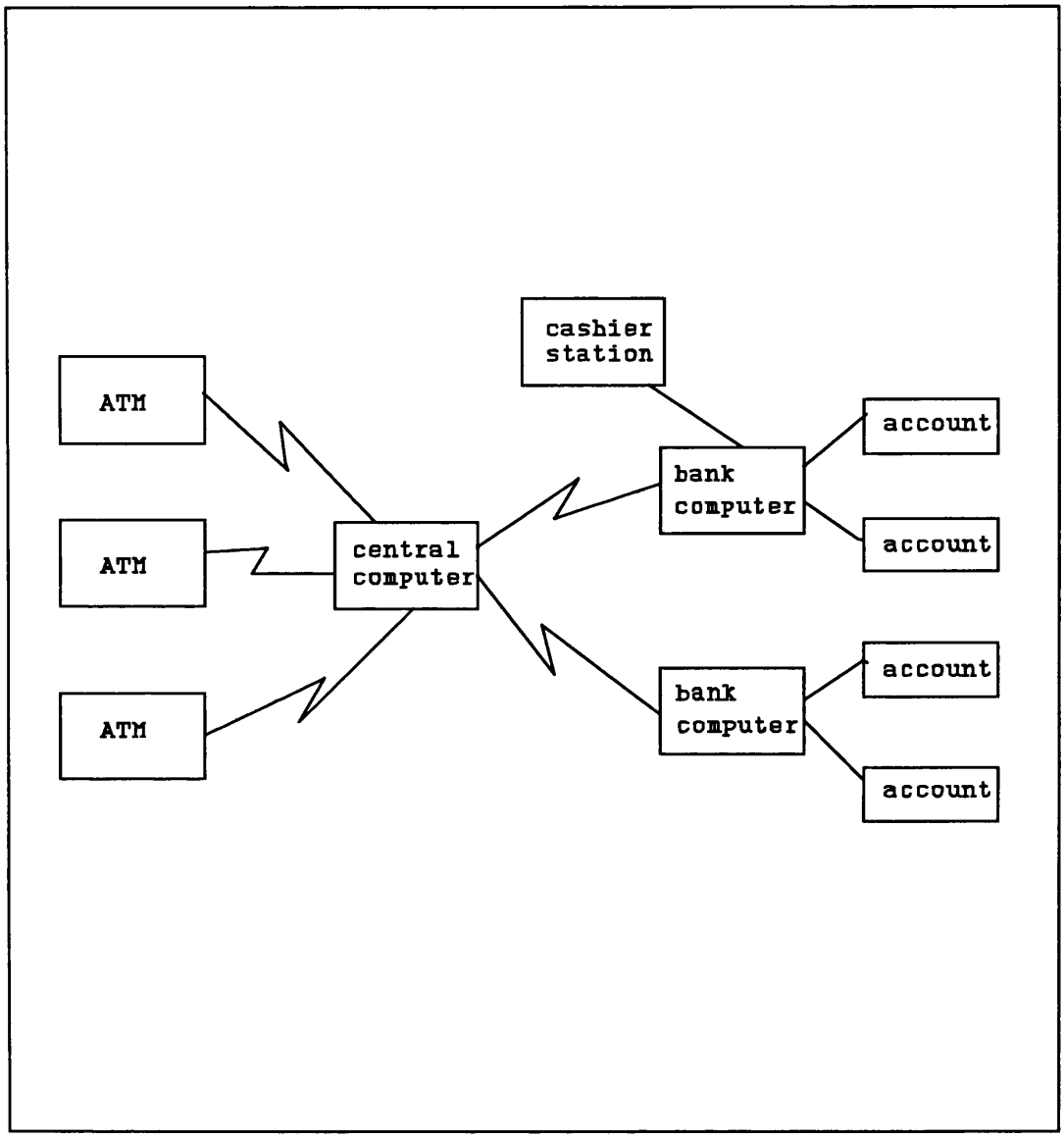


Figure 3.4 ATM network (example adapted from Rumbaugh *et. al.*, 1991)

with a central computer which clears transactions with the appropriate banks. An automatic teller machine accepts a cash card, interacts with the users, communicates with the central system to carry out the transaction, dispenses cash, and prints receipts.... (for a complete explanation of the example refer to appendix I)

Recasting the example into the semantic constraints

We can take this domain of activity and beginning to analyze it semantically. The analysis starts from the collection of relevant material which defines the problem situation. The above description provides a good starting point for analyzing all actions in terms of their existential and ontological requirements. Now we need to separate the

semantic units and put them in a list of agents and other affordances. Each affordance indicates a possible behaviour that is open to an agent. In this context we can identify a number of legally responsible agents:

- List of agents

We begin by listing candidate agents found in the written description of the problem. The following list of agents as the most important semantic units can be drawn as the first result in the study which may be neither completely correct nor complete.

Bank
Consortium of banks
Human cashier
Users

According to the philosophy of semantic agent-based modelling formalism there is no reality without an agent, and the agent construct his reality through his actions. So the ontology intimated here is that of a socially created world. The notion of agents as special type of affordances who are holding responsibility is paramount in semantic analysis. In analyzing the substantive prime tasks of the business, we need to know who is legally or socially responsible (in organisational terms) for any particular portion of business activities.

Representation of agents

On the ontology chart (the modelling representation of semantic analysis) the agent is entered as any other affordance except that we indicate it with an underline so as to distinguish it from all other affordances.

agency structuring

The name of knowing agents should be that in normal usage in the organisation, but we should avoid of using role names instead of the root names of different agents. So instead of user or customer, we should have been more general by saying all persons and all organisations may have

some kinds of relationship with banks. So we refer to **legal person** instead of **user** as the name for responsible agent in our analysis. Legal person stands for any person or organisation.

- Pseudo-agents

An agent can take responsibility and so in this respect an agent differs from other affordances. As in every case the notion of holding responsibility is paramount, agents are in principle human: a person or a legal person like companies. A computer or any other machine like that, for example, cannot be held responsibility in this legal or social sense, and therefore will never be a responsible agent. Machine (in any form: computer, ATM, cashier station, ...) affords formal systems. However, formalism tends to veil responsibility by making it possible to shift the onus onto a machine. Any formalism simply embodies the values of the formalizer (Liebenau and Backhouse, 1990). ATM is a machine governed by rules developed by a responsible rule-giver. Behind any machine which acts as a formal system are responsible agents who diminished their responsibility through applying rules. In this respect machine can be seen as a pseudo-agent which is restricted by rules instead of responsibility. In our example we can list pseudo-agents as follows:

Computer (central computer, bank's computer)
ATM
Cashier station (implies a computer terminal for each cashier)

On the ontology chart the pseudo-agent is entered as any other affordance except that we indicate it with a double underline so as to distinguish it from all other affordances and agents.

Not all semantic units are explicit in the problem statement; some are implicit in the application domain and our general knowledge. Sometimes, we can understand that a certain way of behaviour is open to more than one of the recognised agents.

Generic-specific relationship

Generic-specific relationship between different affordances comes from the assumption of hierarchical groupings and **inheritance** of properties. Specifics inherit all properties of their generic. Figure 3.5 shows the ontology chart representation of generic-specific:

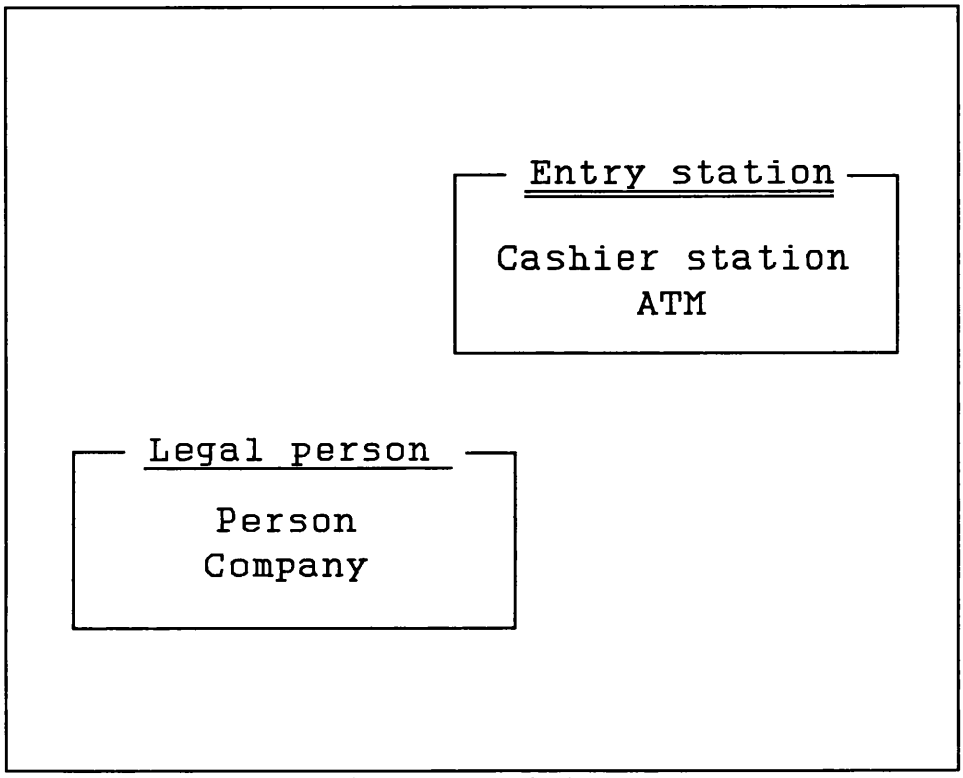


Figure 3.5 Generic-specific relationship in ontology chart

We can discover inheritance from the bottom-up analysis by searching for affordances with similar patterns of behaviour and determiners. In our example ATM and cashier station have the same pattern of behaviour and function. At the fundamental level, they have common structures of communication which can be defined for a generic affordance, i.e. Entry station. They are subclasses of a super-class. This concept broadly resembles the concept of sets and subsets from mathematical set theory. Now a marvelous economy of expression is possible here. Generic-specific concept is also clearly defined in the other modelling formalisms as generalisation abstraction for sharing similarities among different entities and classes. It is also sometimes referred to as or-relationship which means a relationship between an entity and one or more refined versions of it.

- Universals and particulars

When considering the affordances in any domain we need to be able to distinguish between the universal and the particular affordance. The universal affordance connotes some invariant pattern of behaviour in a domain without referring to any specific occurrences, whether actual or hypothetical. All responsible agents are particulars to whom we can refer to particularly. Most of the other affordances in any model will be universals. In our example bank, Human cashier, legal person, entry station and computer are particulars which we can refer to instances of them employing some naming or coding device. But the notion of ownership of a computer by a bank is an example of an universal affordance. A particular is always indicated in the ontology chart by the device of adding a hash mark (#) after it. Where the affordance appears in the chart without any hash mark, then it signifies a universal, whose instances cannot be determined by unique identifiers, such as ownership.

Individuation and identification

As we discussed above, particulars are called identifiable, if there is a general form of measurement, or determination. Determiners serve to distinguish between instances of a universal affordance using different criteria, i.e. different forms of measurement. These measuring systems can be very abstract to be used for picking out any particular *particular.* "Name" is a determiner of any legal person. It is also an affordance and is ontologically dependent on the legal person which permits us to compare and realise one instance of legal person with the others. A hash mark behind an affordance indicates that the affordance is a determiner for its antecedent affordance. Particulars which can be measured and taken apart from a set can be *individuated.* When recording facts about particular individuals, we use the notion of individuality which presupposes that we are able to recognise the uniqueness of each discrete individual. In the following figure (figure 3.6), name represents the determiner of the legal person as an individuated affordance. Agents and pseudo-agents can usually be individuated by some determiners.

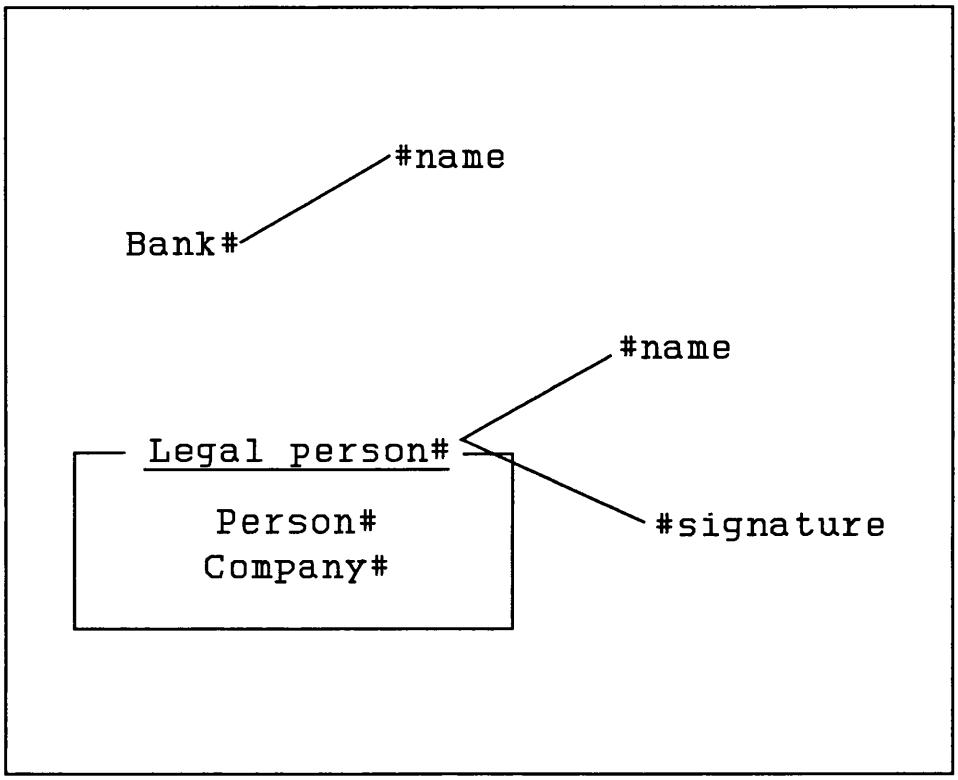


Figure 3.6 Affordance determiners in ontology chart

Real world determiners can help us to refer to the referent of a particular sign in reality. According to semiosis each sign refers to something in reality for **somebody**, so the determination process, employing a system of norms for measuring and comparing different referents of sign is always subjective and dependent on the interpretation of agents. A signature is a formal determiner of a customer for dispensing cash from his account with the bank, while it is possible that a human cashier can uniquely determine a permanent customer of the bank without referring to his signature. Here the cashier applies different set of norms as a measuring procedure to determine the identification of a customer. So it is not always possible to model the reality with a complete set of determiners. Modelling the referent of each sign in reality, without understanding the notion of interpretation power of different agents, would be incomplete.

The notion of real world determiners of an affordance is also known as attributes in other formalisms. Attributes can have values for each instance. Unlike other formalisms with confusion between attributes and objects or

entities, in ontology charting determiners are treated as other affordances which always have a unary association with an individuated particular. We should not confuse these real world determiners with internal identifiers which some implementation media, such as databases, may require to have a unique identifier, i.e. ID number for a person. Internal identifiers are purely an implementation convenience and have no meaning in the problem domain. Explicit identifiers are not required in an ontology chart.

- List of other affordances

In semantic modelling all those elements that constitute the structure of a particular environment are affordances. Agents are special type of affordances, in that they have the ability to realise other affordances. In particular they are capable of taking responsibility for determining the existence of any affordances which are realised. The analysis so far indicated a complete list of responsible agents and pseudo-agents who constitute the most important part of semantic analysis. Now we are in the situation to define a complete set of agents or pseudo-agents for the problem description. These are the focal part of our analysis. Since these agents realise other affordances the tendency will be for them to be found on the left hand side of the ontology chart, they will be able to realise the affordances to the right.

It is necessary to mention that for any domain there will usually be a **root agent** who is the source of all the affordances. Depending upon the extensiveness of the analysis, the root agent might be the business or organisation, the state or society in general. Ultimately all the realizations are traceable back to this root agent (Backhouse, 1990). Figure 3.7 represents the result of analysis so far:

As we can see in the following semantic schema, state (or better to say intentional community) as a root agent is precondition for the existence of all the other affordances. They are called ontological dependent on state as their antecedent affordance.

Until now we focused on agents as unit affordances. Now we can continue our analysis with so-called joint affordances: Affordances which can only exist during the period that

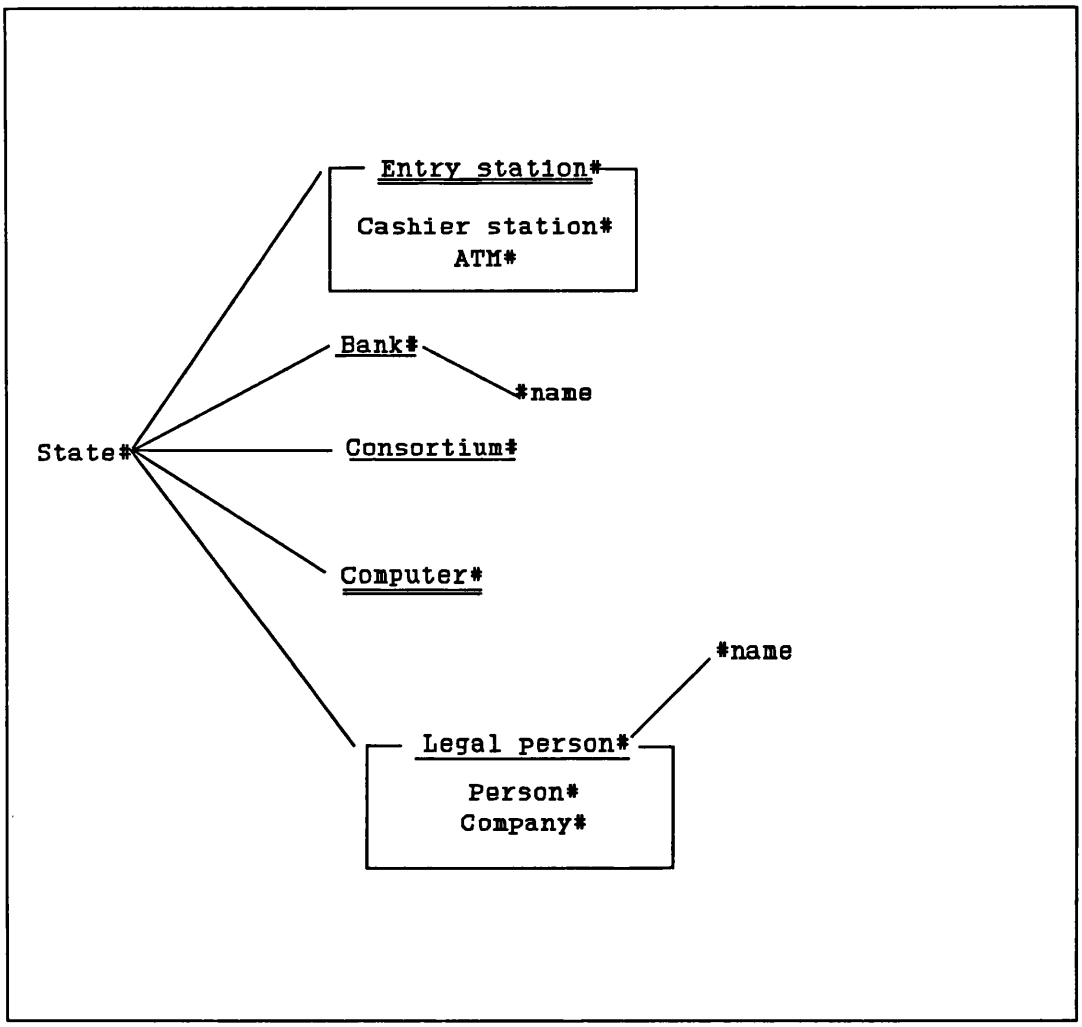


Figure 3.7 Root agent, agents and pseudo-agents in ontology chart

its both antecedents co-exist. It is also important to mention that each affordance in semantic analysis can have no more than two antecedents and if we found an affordance which we think depends on more than two affordances as its antecedents, we always can detail our analysis to break that relationship into two or more relationships between the combination of one or two antecedents. If we forget about human cashiers and the account and its transaction for a moment to keep this example as simple as possible, we can find the following verb phrases directly from the problem statement. Some verb phrases are also implicit in the statement:

Verb phrases:

Bank owns computer

Cashier stations are owned by individual banks

Cashier stations communicate directly with their own bank's computer

ATMs communicate with a central computer

Implicit verb phrases:

Consortium operates central computer

Consortium owns ATMs (consortium shares ATMs)

Banks are members of consortium (a consortium of banks)

The following ontology chart shows the result of semantic analysis so far. This ontology chart of the domain presents itself as a semi-lattice-like structure. Because of the difficulty of representing the complete analysis, it is usual to deal with smaller subsets at any time which concern more restricted spheres of business activities under investigation. Each subset is a cluster of the system under study and different clusters are just naturally connected to each other via common affordances. The **clustered modelling** capability of semantic agent-based modelling will be explained later in this chapter (subsection 3.5.4). As a result of clustered modelling it is possible to gradually develop an enterprise information model of the entire business.

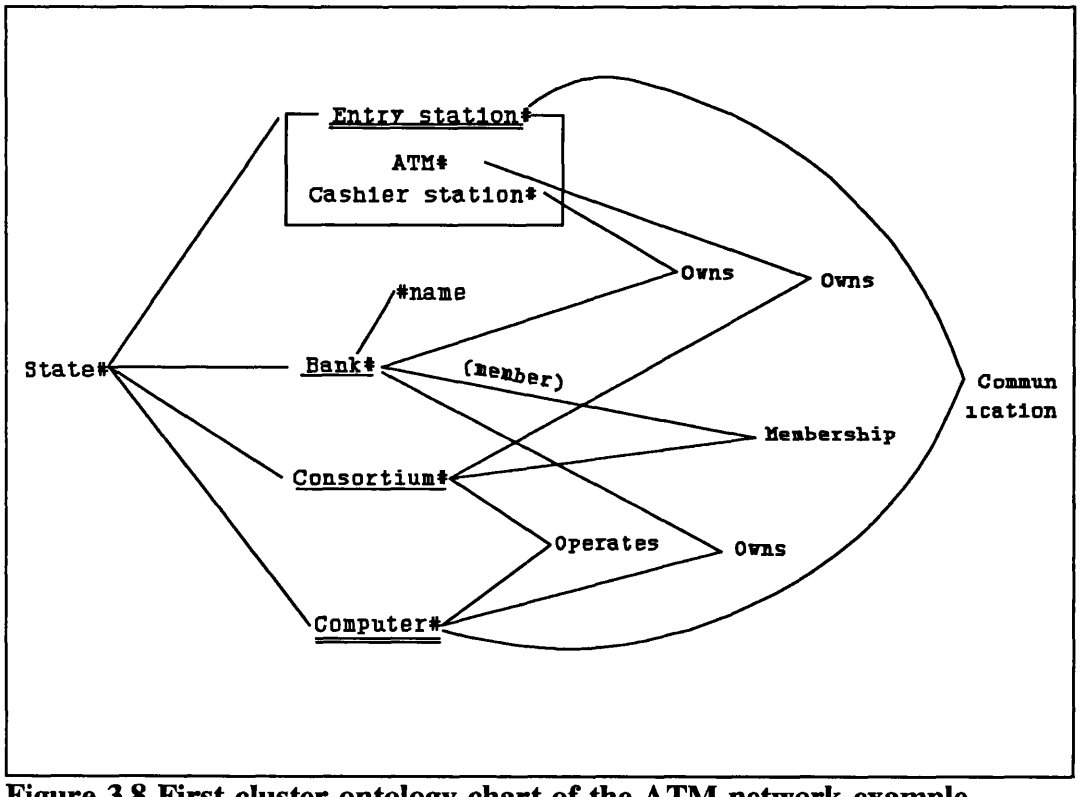


Figure 3.8 First cluster ontology chart of the ATM network example

As shown in figure 3.8, all joint affordances are universal which connotes the invariant pattern of behaviour in the domain of interacting agents and pseudo-agents, without referring to any specific occurrence. This usage is similar to that of entity occurrence in the Entity Relationship modelling or the assumption of association between object classes in object-oriented modelling.

The ontological dependency relation states that the antecedent precedes the dependent. The dependent depends upon its antecedent for its very existence - if the antecedent should cease to exist then the dependent no longer exists. A whole chain of dependencies can arise with predictable cascade effects should one of the antecedents cease. Affordances may have either one or two antecedents, and where there are two, both must exist contemporaneously for the dependent to exist (Backhouse, 1990). So through these chains of dependency, it is possible to build complex networks. As we can see in the following simple ontology chart, lines of dependency connect the dependent affordance with its one or two immediate antecedents. Only the root agent (the first affordance on the extreme left of the chart) has no antecedents. For most analyses the line may be drawn at the level of state as the root agent. The graphical notation we use for describing the dependency networks make use of a strict left to right order. A term connected to another term to its left, depends on this left term for its existence.

Roles: modified agents

Often roles arise while a relationship exists, so that during its existence a special role name applies to either or both of the antecedents; a bank participating in an ownership relationship has the role name of owner, whilst a computer as an asset in the same relationship might be referred to as property. Roles are the special behavioral possibilities for every particular affordance which is the antecedent to a dependent. The particular that occupy the role is called role-carrier or **modified agent**. Not every relationship we define has specific roles, but when necessary, making use of role names can be highly expressive and economic. The role is an invariant way of behaviour of a particular or individual agent which should

not be confused with the *role name* which is the name that we use to refer to the role. Role names are subject to change and are useful to modify the representation of a particular affordance or agent in a particular course of action. Representation of roles in the ontology chart requires no special treatment. Where there are relationships in which roles apply, the role name is written in parentheses along the ontology line that connects the antecedent to the dependant. The role name applies to the antecedent. The use of role names provides a more readable ontology chart through representing modified agents in addition to agents or pseudo-agents. In the above ontology chart shown in figure 3.8, bank and consortium afford membership, and each particular bank in this relationship carries a role so-called member. Member is a modified agent name for bank in that specific relationship.

Role qualifiers

According to the definition of roles, new ways of behaviour open to a modified agent as a particular affordance, the modified agent on its turn may have affordances and also sometimes needs qualifier. Role qualifier is ontologically dependent on the modified agent and can uniquely qualify a particular role-carrier. For example, let us suppose that in the above ontology chart membership of each bank in the consortium can be uniquely qualified by a membership number as bank code. So bank code is a qualifier for which its immediate antecedent is member as a role, although in any case, the ultimate antecedent is, in fact, bank. The qualifier is a special type of affordance that reduces the effective multiplicity of a role. Consortium may have many banks as its member, but bank code distinguishes among different banks every member in the consortium. Consortium members can be qualified in this way. Role qualifiers improve semantic accuracy and increase the visibility of navigating paths.

As computer systems are the main storage device for recording and retrieving data, role qualifiers are the most popular access code for different

agents when performing different roles. In this respect, role qualifiers refers to a specific record in a storage system instead of something in reality. So role qualifiers are similar to the functioning of keys in database systems. As increasingly, this analogy is going to be widely used in our real life, role qualifier can now uniquely qualify each modified agent in a specific role (but it is not necessarily able to clearly identify its associate agent, as their measurement procedure for qualification is restricted to a specific role of an agent instead of himself. Still the problem of relating any modified agent to its antecedent agent -role carrier- exists). Role qualifiers and specifically numerical codes are very popular in reality as technology surrounds our daily life. Nowadays, every person may carry different roles with different qualifiers, i.e.; library membership no., employee no., student no., driving license no., etc. It is important to reiterate that agent determiners show what is the referent of a sign in reality, but role qualifiers can only point to the referent of a specific instance of a particular sign in reality in a specific role and mainly for the purpose of computer technology. Role qualifiers cannot refer to the general referent of a sign, but they refer to a particular instance of a class affordance in a specific role. Name is the determiner of a person, but his employee no. might just qualify him as an employee of a specific company and may not be useful in qualifying the same person as a driver or a local library member.

Role qualifiers are always shown with a hash mark behind them. Role qualifier and its associated modified agent in the ontology chart in an abridged form are shown together in the parentheses with an undersign between them as their separator, i.e. (member_#bank code) or (communicator_#station code). The undersign represents the ontology dependency of role qualifier to the role.

Ontology chart in figure 3.9 represents a refined **first cluster** ontology chart for the problem statement in the example based on the analysis so far:

As we can see in the above chart, station code is qualifier of any particular ATM or

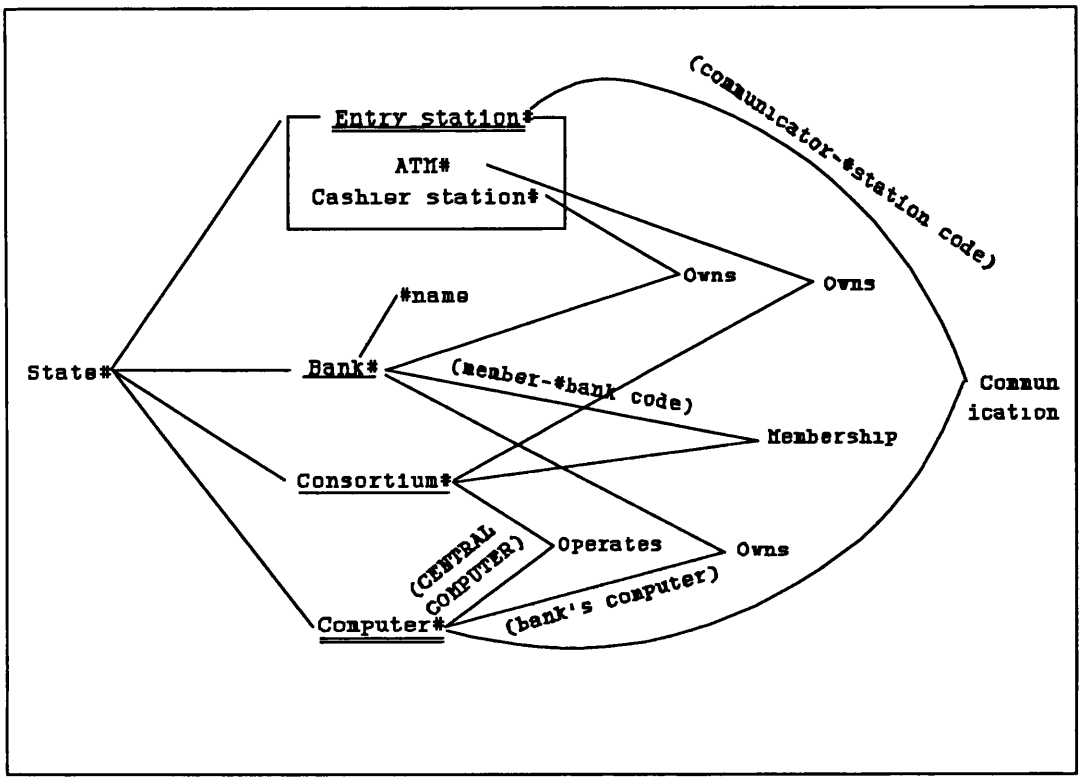


Figure 3.9 Role names and role qualifiers in ontology chart

cashier station in their communication with computer and bank code is also the qualifier of any member of the consortium.

Now that we had one cluster of ontology chart, we can focus on the other parts of problem statement about human cashier, account and etc.

- Whole-part relationship

The concept of generic-specific is also valid for cashier. This specific name represents a responsible agent who gains his authority from the constitution of the bank where she or he is employed. This comes from the knowledge of problem domain that bank employs cashiers and that he is an authorised transaction handler in banking system. So banks consist of a set of specific posts -generically known as positions- based on their organisational structure. Cashier is a specific post under the generic affordance such as position in the bank. Each position may be occupied by different agents who are responsible for that position and should have employment relationship with bank. Person with the employment relationship with bank has a role name of employee which is also qualified by an employee code. Position is not an independent affordance and

has a *whole-part* relationship with bank. It is a generic affordance as part of a larger complex agent, banking organisation, which belongs to that certain whole. A part-whole relationship is also an ontological relationship: a part cannot exist without the whole. Notation for whole-part dependency in ontology charting is a line as a normal ontological dependent affordances but with the addition of a large period to represent the subdivision. So we denote them in the following way (figure 3.10):

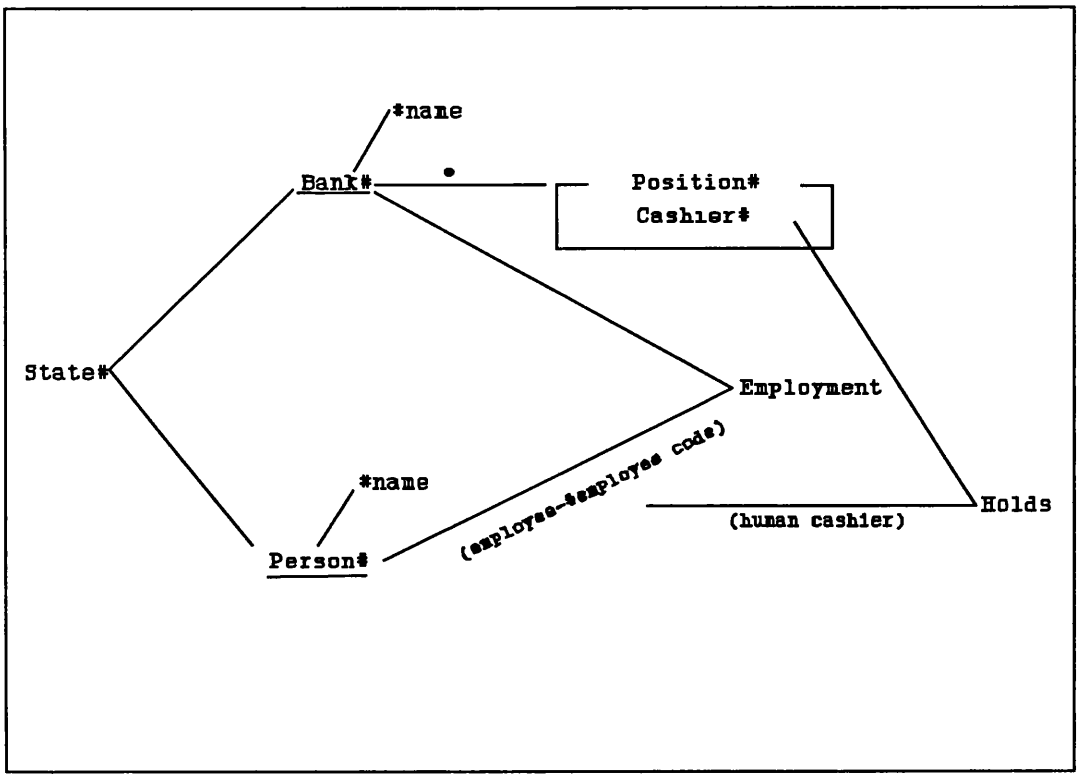


Figure 3.10 Whole-part relationship in ontology chart

Figure 3.10 represents the **second cluster** ontology chart of the domain under study which explicitly deals with the meaning of cashier within the context of banking organisation.

The concept of whole-part dependency is similar to the treatment of aggregation in other modelling formalisms like object-orientation. An aggregate is made of components and components are part of the aggregate. An aggregate can be semantically treated as a whole unit in different purposes, although it is made of several lesser parts. Whole-part relationship is also sometimes referred as an and-relationship in different modelling concepts.

- Sign types and communication acts

The other verb phrases directly from problem statement and also some implicit verb phrases related to them are:

Verb phrases:

Cashier enters transaction for account

ATM accepts cash cards

ATM interacts with user

ATM dispenses cash

ATM prints receipts

Bank computer processes transaction against account

Central computer clears transaction with bank

Implicit verb phrases

Customers have account(s) with the bank

Cash card accesses accounts

Figure 3.11, **third cluster** of ontology chart, shows that account cannot exist unless there is a contract between bank and a legal person to comply the rules of governing that account. In this relationship legal person has the role of customer. For each separate contract with the bank, customer has an account with the bank. In this relationship customer has the role of account holder and will be qualified in this role by an account number.

Analysis of the above verb phrases show that there is a transaction as the result of interaction between customer and cashier (and then between cashier and cashier station) which we recognise it as **cashier transaction**. Also there is another type of transaction as the result of interaction between user (or better to say cash card holder who might not necessarily be the customer himself) and ATM which we recognise it as **remote transaction**. Transaction is defined as a single integral **request** for operations on the accounts of single customer. In order to have a better understanding of the meaning of transaction, let focus on a specific instance of a transaction in real situation like withdrawal. The transaction is expressed in the form of requesting for withdrawal

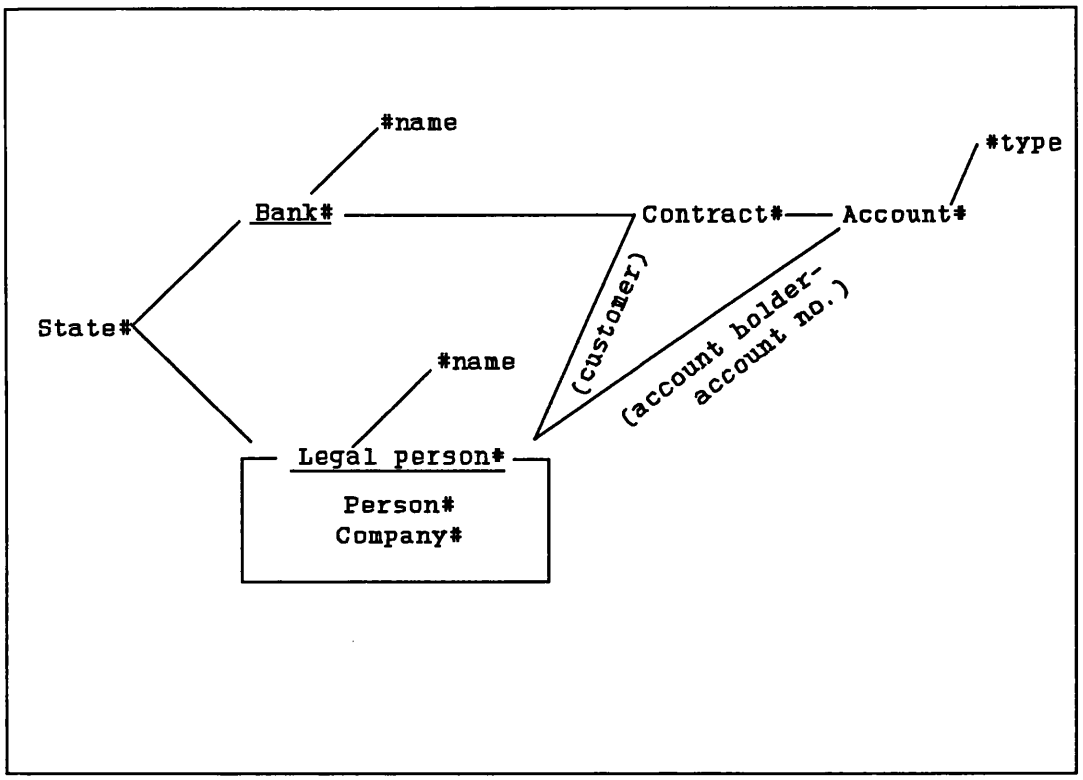


Figure 3.11 Ontology chart for an account within the banking system

certain amount of money from the account of an account holder. This begins with a communication act by a responsible agent. If we are going to analyze this communication act, first we must consider what the necessary and sufficient conditions are to determine whether an act of requesting for withdrawal money has to be performed in a particular uttered request. We can identify a set of propositions which, taken together, specify that an agent made a request for withdrawal. So each condition will be necessary condition for the performance of the act of withdrawal and, taken collectively, the set of conditions will be a sufficient condition for the act to have been performed. Searle (1986) holds that a collection of some general rules and some specific rules are needed to take care of those conditions. So here is how we specify the conditions which constitute the rules for withdrawal money:

General rule 1: Normal input and output conditions obtain

This rule is intended to cover the conditions such as the use of a mutually intelligible language, that the conditions of communication are not extraordinary, and that those involved in the communication are prepared seriously to cope with the kind of request to follow. (for example the

request for withdrawal money is not a request made by an armed band of robbers)

Specific rules:

These rules mainly deal with substance of the request and its content. They identify the expectations of the people involved and approve that the request is genuine. They must also provide the understanding that the uttering of the request will oblige the customer to stand by his request, so it might be accompanied by a signatory obligation. These are specific strict rules for changing the obligations of one agent to another.

General rule 2: seriousness of communication act

The requester intends that the utterance of the request will produce a belief that all the above conditions obtain by means of the recognition of the intention to produce that belief, and he intends this recognition to be achieved by means of the recognition of the request as one conventionally used to produce such belief. This rule might be regarded as a part of general rule 1, but it further explains what is meant by the communication act being of serious intent.

General rule 3: understandable semantical rules of communication

The semantic rules of the dialect between two parties in the communication act are such that the request is correctly and sincerely uttered if and only if all the above conditions obtain.

This form of analysis gives us an opportunity to see in great detail how the pragmatic character of a situation can be understood. We can see how a careful analysis of the communication act in a conversation can provide us with an understanding of the intentionality of the participants. Each transaction within the banking system means an exchange of legal obligations between parties and needs to be considered very carefully. Specific rules of the communication act must be formalised as well as the general rules. This gives the banking system a firm basis on which to process the transaction without

ambiguity. In order to formalise these rules which are mainly concerned with the pragmatic properties of signs, we need to understand semiological (linguistic) affordances. Semiological affordances are affordances that stand for realisation, and provide other derived affordances. This gives us the understanding of the contextual framework within which communication takes place, therefore we can build a firm basis to approach the semantic properties of signs.

In semantic analysis there are two related kinds of semiological affordances encountered in communication acts: **sign tokens** and **sign types**. A realised instance of a sign is a sign token, such as a cash card or piece of cheque completed by an account holder, whereas the pattern of the tokens are types and are realised as abilities to interpret the token. For information systems analysis we are usually interested in sign types; the sign tokens are the interest at the empirics level. In performing semantic analysis we are concerned with discovering the meaning of the signs. To formalise the realisation of a semiological affordance (a request for withdrawal of money), banking system requires to apply very restricted interpretation of a complex sign types in a form of a cash card (plus its security code) or a written request like a completed cheque as sign token. In this way bank can imply that the account holder realises a sign type that means he want to withdraw money. He is forced to use specific sign types to communicate his request firmly and without any ambiguity, and to commit transactions. Agents need to make special use of sign types when performing communication acts with bank. When transacting business, frequent use is made of communication acts: requesting withdrawal, asking for deposit, acknowledging money transfer between accounts and so on. In every case the agent must use a sign token when effecting the communication act, where the sign token has its referent as some pattern of behaviour.

Rules of communication acts

Communication acts have two antecedents: one must be a responsible agent and the other a sign type. An agent realises a sign type that has a specific meaning (interpretation). He uses the sign type to communicate and assert his action. If we wish we may analyze in detail the method of

communicating - the form of the sign token - whether speech, writing or body language. For semantic analysis it suffices to be able to represent the meaning of the sign.

Representation of communication act

Figure 3.12 shows the general representation of communication act in ontology charts. The general representation of communication act implies that communication act has two antecedents: an agent who uses a sign when effecting communication act and a sign type representing some pattern of behaviour. Sign types are ontologically dependent on the agent to represent his intention. A sign type is represented within quotation marks and also underlined. Examples of sign token can be given under the underlined sign type. For example at the high level "request for money" is a sign type. At the lower level spoken request at counter or use of cash card at ATM is a sign token.

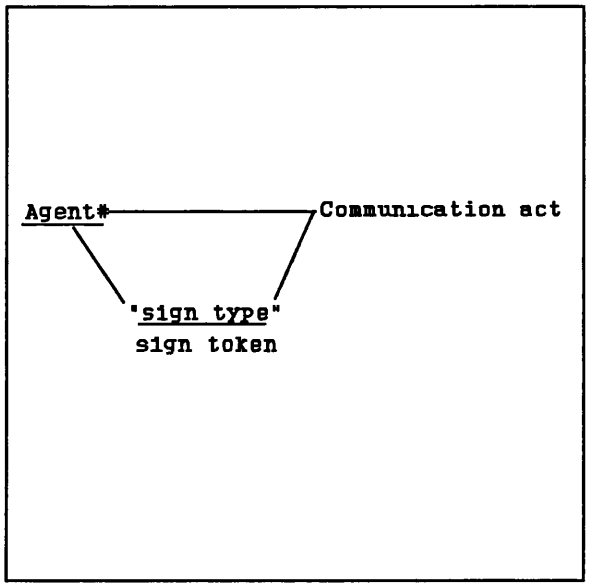


Figure 3.12 Communication act representation in ontology chart

The following figure (figure 3.13) represents the ontology chart for transaction.
As it shows request is a communication act by legal person which is interpreted by a pattern of behaviour expressed in a sign type like "transaction sign". "Transaction sign" uses sign tokens like cash card or cheque. Transaction will be produced as the result of this communication act in the form of a request and based on its type (cashier transaction or remote transaction) will be entered in appropriate entry station.

Using sign types also enables us to use the notion of times (Albadvi and Lee, 1996).

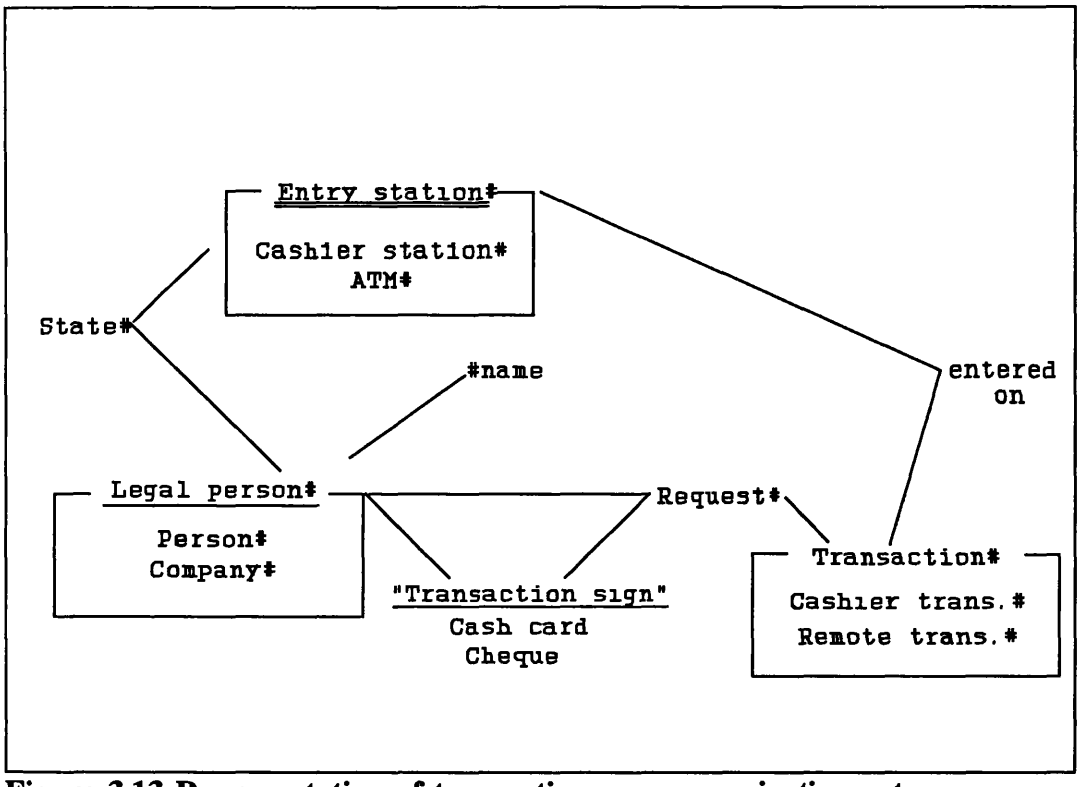


Figure 3.13 Representation of transaction as communication act

Based on this type of semiological signs, time is not part of the objective reality. We can distinguish between what exists in the here-and-now and what belongs to the world of signs. All the other reality, whether past, future, distant or just mere conjecture, has to be constructed from parts of agent's present reality. When an agent is able to carry forward memories from the past to the present, this is similar to a semiological ability. Sign types can be used to bind together events at different times (Backhouse, 1990).

Authority in communication act

An agent is said to have authority for matters for which he can take decisions. When we have a complex sign type representing a rule, realisation of the sign type is an interpretation of rules made by a rule-maker. Therefore uniform interpretation of a rule like "transaction sign" as a complex sign type in a communication act must be the responsibility of an agent as rule-maker. So every sign type must have an authority as a component to interpret it. In our example bank authorities are rule-makers for "transaction sign" which is used in the form of sign tokens, i.e. cash cards, by customers. The representation of authority for sign types in

ontology chart is by means of @ sign after the sign type, followed by the responsible rule-maker for that sign.

The following figure (figure 3.14) represents the **fourth cluster** of ontology chart of the domain under study which also completes our semantic analysis for this problem.

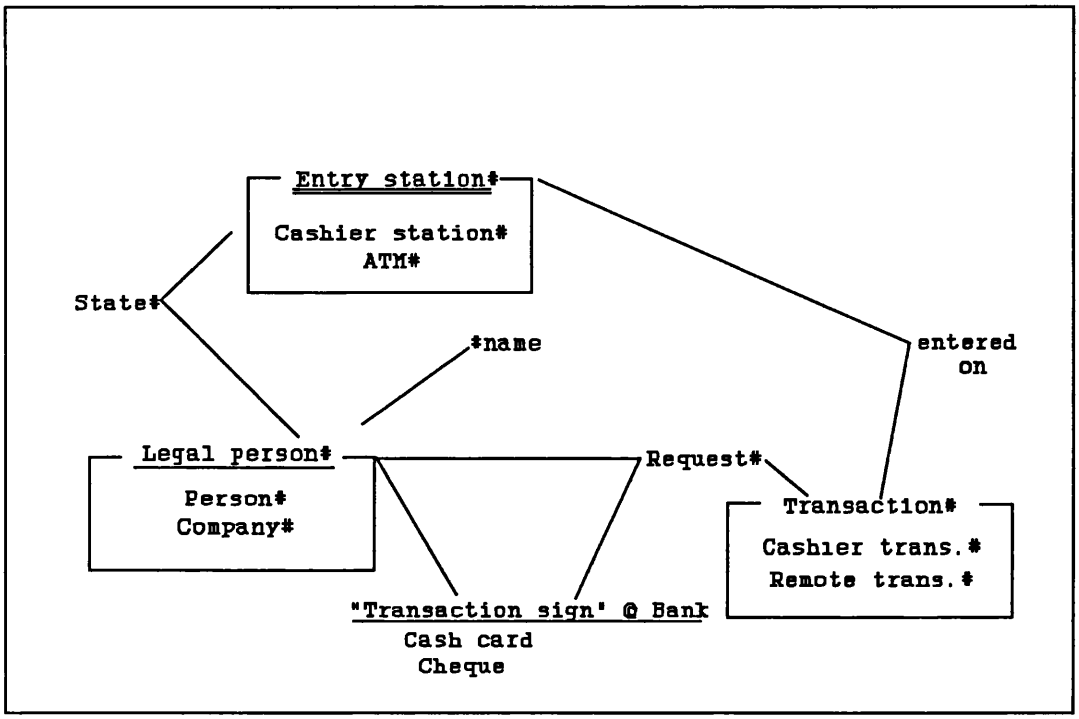


Figure 3.14 The notion of authority in communication act

Implicit features of ontology charts

There are two important implicit feature at every node of an ontology chart. It is necessary that we mention them here before closing this subsection. They are recognised as (Backhouse, 1990):

- **Start and finish times**

The ontology chart shows how affordances depend in their turn upon the existence of their antecedents; time is a universal parameter measured in terms of the existence, from start to finish, of any affordance. **Surrogate table** is a device to record antecedents of any dependent affordance. In this table there are also two separate columns for the start date and finish dates

of each affordance. We will discuss the surrogate table fully in charter 8 of this thesis. Part of the task of the semantic analyst is to attempt to complete the entries for start dates and finish dates in the surrogate table. For universals there will be great difficulty in giving start dates, but for particulars, it is less problematic. Particular persons have their birth dates and these are their start dates. Where we do not know this information we are continually confronted by the blank entry that reminds what we must yet discover. When searching for this information we have the aid of knowing who is the responsible agent for determining the start and finish of each affordance. This leads to second feature of ontology charts.

• **Agents, responsibility and authority**

Realising an affordance must be the responsibility of an agent who has the authority to determine whether it exists or not. In the simplest of cases this is when the agent uses his judgment to decide. Responsibility is created when decisions are made. If nothing is decided, there is no responsibility. An agent is said to have authority for matters for which he can take decisions. So authority is different from responsibility, the authorised agent has the freedom to take the decision or to just leave it. Responsibility is applied authority. Every affordance in ontology chart must have an authority as a component. This is sometimes (if the analyst thinks it is necessary) shown by an @ after the affordance followed by the name of the authorised agent. But it is necessary to have the name of responsible agents in surrogate tables in separate columns for each affordance. For each of the affordances that populate the domain of our analysis, we need to know who takes responsibly for deciding the existence of each affordance. In some cases there will be rules or norms which govern the process of deciding the start or finish dates of some instance.

It is possible to transfer authority. Very often in organisations, decision-making is delegated, because the agent cannot possibly handle all the decision making on his own. Very likely because of geographical reasons

or of insufficient capacity. When authority is delegated, attempts are made to make the decision making more uniform across the several subordinate decision makers. For this purpose, norms can be used. The original authority makes a norm (a complex rule) and is responsible for the contents of this norm. But even here the rule will eventually have to be interpreted by some person or persons. For any particular realised instance there must have been an agent responsible for interpreting the rule made by rule-maker. It is possible to have different responsible agents for the start and for the finish. In the context of our example about bank account, a bank clerk may have responsibility for the opening of an account whereas closing an account might be done only by the express permission of the manager and customer both.

Realisation of instances of each affordance class in ontology chart is described by norms. Norms are a way for describing the rules about when affordances start and finish. Starting and finishing, together with responsible agents to realise them, are the only things that can happen in semantic analysis, but using these two possible events, very complex norm structures can be built.

3.5.4 Subject area Clustering concept

Subject areas, sometimes called semantic collections, are an important complexity-reducing concept of semantic analysis. Subjects are particularly important to rapid development of systems because of strong enhancement they provide for analysis reuse. Reused ontology charts, when they are well developed and have been implemented in a powerful CASE (Computer Aided Software Engineering) environment with the capability of easy modification, make excellent *free* semantic components for reuse in new systems.

Reducing complexity through subject clustering

The information model of most real life systems contains hundreds of entities or object classes. Such models will be very difficult to manage on a computer screen using CASE tools, because designers will be able to view only a small section of the model

at any one time. Reviewers and users looking at such models will find them inaccessible and intimidating. There will be difficulties in handling such models from analysts to users and designers, designers to programmers, and developers to maintenance programmers.

The structured approaches were, however, very good at managing specification complexity through hierarchical decomposition of dataflow diagrams. This was both good news and bad news for rapid prototyping of system clusters. The good news was that each page of a hierarchical set of dataflow diagrams could be presented in an easily understandable form on standard notebook size paper. The bad news was that modifying the hierarchy of pages during prototype iterations was unproductive and was a configuration management nightmare. Other than the context diagram (dataflow diagram level zero), where system external interfaces were specified, and primitive level diagrams, where the actual functionality was specified, the middle levels represented only an artificial packaging. Dataflow diagram packaging involved complex rules for balancing, levelling, and partitioning of dataflows and processes, work that had to be modified when system functionality changed during prototyping. CASE tools are helpful in reducing the amount of effort required to do this rework, but do not eliminate it; semantic analysis does.

Subject clustering presents a nice compromise between the inaccessibility of large, flat, object-oriented information models and the unwieldy, difficult to modify, long-legged levelling of elaborate dataflow diagram decompositions. A subject area is a collection of ontologically bounded affordances that all relate to the same general area of the user work situation. They are semantically related and as a collection convey an integrated meaning of a substantive subject. Ontology charting can be reviewed one subject at a time. An analyst working with a CASE tool can work on one subject area at a time. Small teams of programmers can prototype single subject area incrementally or concurrently. The printout of a single subject area will probably fit on most desktops. There are no arbitrary rules (such as seven entities plus or minus two) for the maximum number of affordances in a subject area as with structured approaches. Subject areas do not create balancing and partitioning problems for rapid prototypers. Balancing is

not important in ontology charting, because data does not flow here, it is signified by rigid signs. Once subject areas have been specified, they tend to be very stable; they do not go away. New affordances can be added to subject areas in prototype iterations, and additions never jeopardise the underlying soundness of the ontology chart. But affordances rarely migrate from one subject cluster to another. This is because subject areas are defined by the semantics of the affordances and their ontological dependencies, an entirely different concept from structured process partitioning.

Analysis reuse at the subject level

Whether units of analysis, i.e. entities or objects, which are common to two different problem domains are a good strategic target for reuse depends on the semantics of their particular subject areas. In other words, whether two object classes or entities with the same name are really the same depends on their semantics and connections with other objects or entities. In semantic analysis each subject area contains a number of ontologically dependent affordances whose connections to each other are strictly bounded by existence limitation. They cannot exist without being connected together and there is no arbitrariness in choosing affordances within each subject area. They are just bound together as the result of their natural ontological dependency. No other analyst can have them differently. Therefore, if a problem domain has an entire subject area in common with an existing problem domain, then all of the existing analysis in that subject area can be reused. As each subject area is a natural outcome of semantic analysis for related affordances, there is no artificial clustering and semantic subject clusters can resolve some of the troublesome issues we currently experience in attaining high degree of analysis reuse. Use of **natural clustering** characteristic of semantic analysis would make for easier comprehension of the analysis reuse target and provide for the possibility of reusable subject area collections stored in a public repository systems within an enterprise analysis level. The following figures (figures 3.15, 3.16, 3.17 and 3.18) represent subject areas for the example in previous section. No special effort needed to define subject areas: they have emerged naturally from the process of analysis and ontological dependencies of different affordances to the main agents recognised in the first analysis.

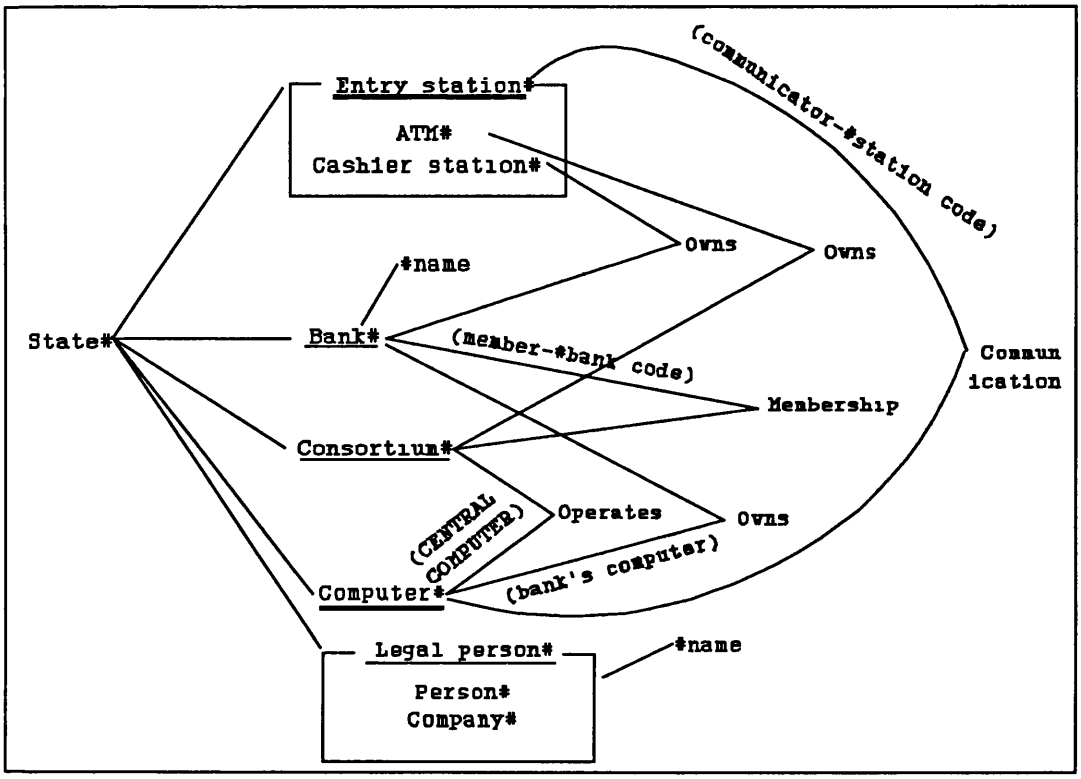


Figure 3.15 Cluster one: Basic communication relationships

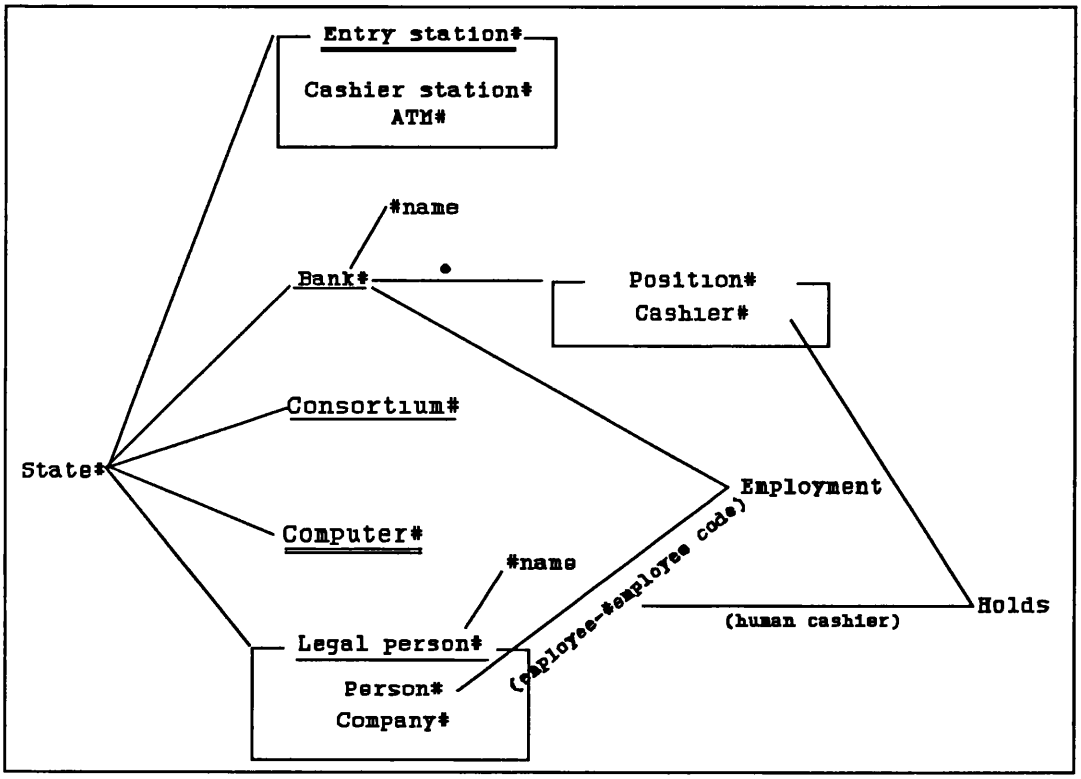


Figure 3.16 Cluster two: Cashier representation

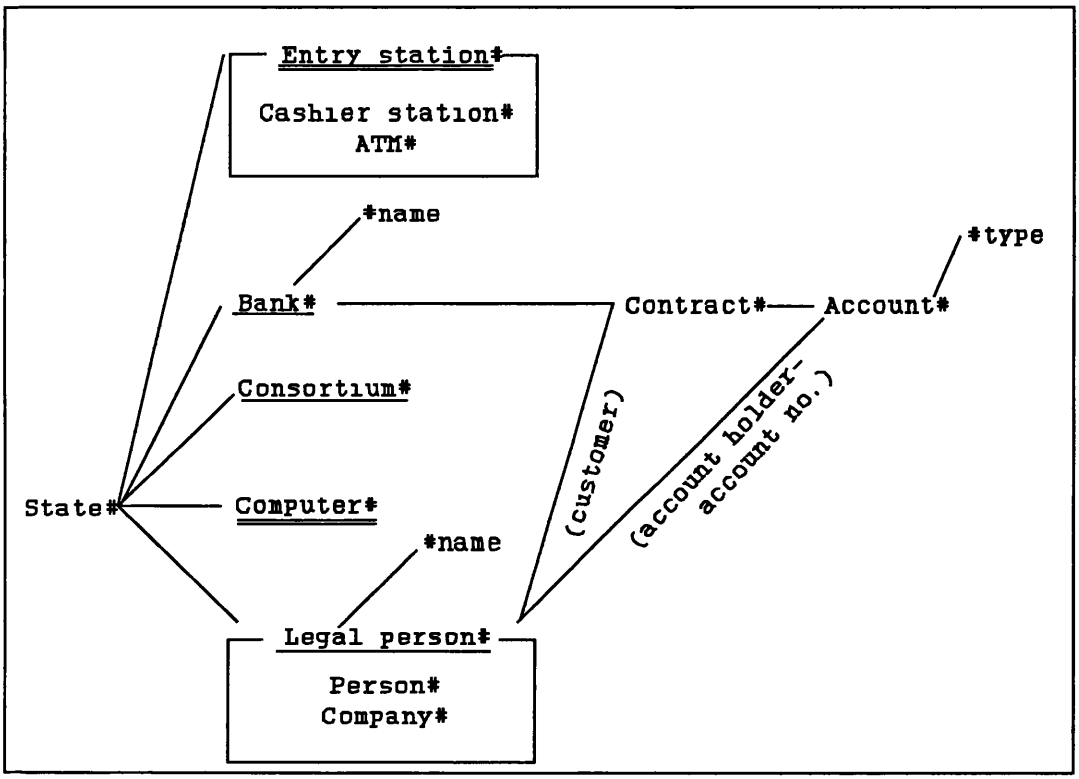


Figure 3.17 Cluster three: Account representation

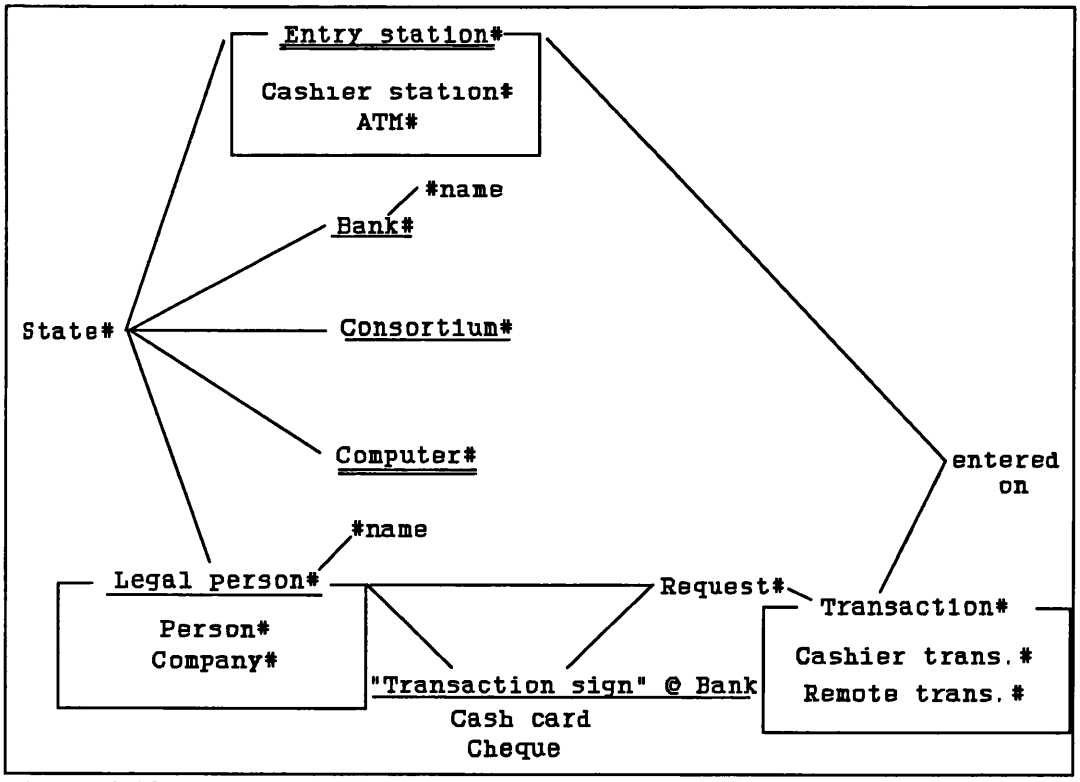


Figure 3.18 Cluster four: Transaction sign type

Guidelines for specifying subjects

Analysis reuse at the subject level will be encouraged to the extent that analysts do a good job of subject clustering. The following are some suggested criteria for subject clustering. The objectives of the criteria are examined one at a time to discover what purposes will be served by modelling according to these criteria.

- **Considering loose coupling of subject areas**

It means minimizing the number of connections between the affordances in the subject area being defined and all other affordances in the model. Limiting those connections to the responsible agents in the most left hand side of the ontology chart.

If affordance classes are mostly connected to other affordances within the same subject area and rarely connected to affordances in other subject areas (except root agent and its immediate dependent agency structure), they are contributing to the semantic definition of the subject area. The whole subject area will make more sense to a browser to the extent that this is true. Coupling always reduces reusability. The more tightly coupled a subject area is to other subject areas within a model, the more difficult it will be to reuse in a new problem domain. If a subject area has a minimal coupling through its affordances in other subject areas within a model, there is a good chance that the entire subject area collection of affordances will be easy to reuse in other problem domains with similar requirements subject matters. Loose coupling reduces the ripple effect where modifying one item may cause defects in others. It also makes system specification easier to follow and understand. The above example demonstrates a very good example of loose coupling of subject areas.

- **Considering the contribution to the semantic definition of the subject area**

The other dimension of the guideline for minimal coupling between subjects is that we need to include an affordance in a subject area only when it

makes a contribution to the semantics of the subject area. It is natural to see that if we have affordances within a subject area that were only included for ontological convenience, the analyst cannot figure out where else to put the affordance. This will also make it easy to name the subject area - the test for violation of this criterion. If analyst has trouble naming the subject area, it will have a reduced chance of appearing on a browsing potential reuser's hit list. It will have a reduced chance of being a good fit in any other problem subject area.

- **Considering analysis for reuse**

The analyst needs to identify subject areas that may exist in any other problem domain. It will not take extra effort to identify those subject areas. It should be second nature to a good analyst. Subject areas that make very specific assumptions about a given problem and have very specific semantics will tend to become isolated in a specific problem domain. Semantic analysis represented in the ontology chart can have a much longer life expectancy than other analysis techniques because of the extensibility and easy modifiability of the chart. Therefore, if the analyst identifies those common subject areas with a chance for reuse, the life of an ontology chart will be long enough to be reused again.

It seems to us that in semantic analysis the comprehensibility of a subject area must be traded off for the elimination of a hierarchical functional decomposition proposed in other analysis techniques. Hierarchical functional clustering, which is mainly handled by syntactical analysis of functions within a problem domain, builds a ripple effect into a system and makes prototype iteration unnecessarily difficult.

3.6 Conclusion

The purpose of this chapter was to suggest a new theoretical framework for evolutionary development approach based on planned organisational change theory and semiotic theory. The semantic analysis technique can be harmoniously applied within

the Lewin's (1952) three-phased model of change process: unfreezing, moving and refreezing. Semantic analysis is introduced as a technique for specifying the information requirements which can support the process of determining meaning and interpretation within the organisational context. Through its formalism - semantic agent-based modelling - and its graphical representation, ontology charting, the analyst is able to produce a representation of the business where the terms used to describe the organisation are semantically normalised, that is, also subjected to rigorous constraints that ensure no ambiguity exists. By specifying the underlying business tasks in this manner, the information requirements of the organisation can be addressed in the form of the invariant patterns of behaviour by responsible agents. It gives a conceptual model of the business which is consistent and less likely to be subject to change. All these characteristics suggest that this analysis technique can be useful in unfreezing and refreezing stages of planned change process.

The semantic analysis technique also has an important complexity reducing concept, namely a subject area clustering characteristic, which makes the technique easily applicable to large scale and complex business problems. By emphasising semantic ambiguities, semantic analysis focuses on those subject areas with greatest uncertainty, hence greatest risk and which should therefore have priority when the prototyping effort has to be limited in its scope. Each prototyping effort can be concentrated on a specific semantic collection of a subject area in an evolutionary development approach. The semantic collection and grouping of requirements with their consistent ontological dependency characteristic suggest a major advance in the effective identification of requirements to prototype.

Semantic analysis facilitates the negotiation of meanings and the mutual understanding needed for the change process in different stages of implementation, from unfreezing through refreezing. It requires that different cognitive styles of different responsible agents mould together a unified frame of reference which will serve as a consistent perceptual filter through which we can interpret organisational activities and information requirements. To achieve a unified frame of reference, we need to bridge the semantic gap between responsible agents. This will ensure that responsible agents are involved

and that they are needed to form a realistic view about the new system. Therefore semantic analysis contributes as a medium for conceptual training throughout the development process, beginning with the unfreezing stage. The unified frame of reference also gives the users the opportunity to view original requirements during prototype presentation. This will allow them to portray the requirements together with prototypes at each iteration of the refreezing stage, which should result in a satisfactory evaluation of prototypes. This is a major step towards cementing the new system in place and to institutionalising the change during each cycle of evolutionary development.

Applying semantic analysis and prototyping techniques within the three-phased planned change process model will allow the developers to keep the system implementation process open to feedback about change process, while also heightening user involvement in the clarification of their roles in relation to the formal system and in negotiation about the extent of their responsibility.

In this chapter, we proposed that planned organisational change theory and semiotic theory are suitable for a new theoretical framework for evolutionary information systems development. The breadth of development activities captured by semantic analysis and the planned organisational change model and the harmony between them is encouraging, but it would be unfortunate if such breadth generated feelings of complacency. There is, after all, a difference between theory and practice. As suggested in this chapter, within the theory arena, we were able to recognise and tie together existing knowledge and theories. Both theories seem to be in peculiar position of complementing each other and offering more together than separately. The theoretical harmony between semantic analysis and the planned change model and their common relevance to the shortcomings of evolutionary prototyping is welcome, yet much more practical guidelines are needed. A stronger grounding is required in both theories to maximize their contributions. The findings of an exploratory case study, conducted at a large car manufacturer, provide more insights into the problems of the evolutionary development approach. The case study, the subject of next chapter, will provide a stronger linkage between theory and practice.

Although the characteristics of semantic analysis technique and the three-phased model of planned organisational change process seem ideally to respond to the shortcomings of evolutionary development approach mentioned earlier in this thesis, we still need to know how to develop a quality evolutionary information system in a business change environment and how to manage it systematically. Therefore, in order to prevent being hampered by the ever present dichotomy between theory and practice - betwéen theories of change and the art of changing - we are required to focus on a methodology to foster the viability and progress of the proposed theoretical framework for evolutionary development. This leads us to a proposal for a new development method in evolutionary approach, the subject of chapter 5.

CHAPTER 4

Exploratory empirical study

Chapter Overview

This chapter presents the findings from a case study which identifies constraints encountered when implementing an evolutionary prototyping approach in a business change environment. The exploratory case study investigates the identification of the main difficulties of the evolutionary development approach in practice and relates them to the proposed theoretical framework presented in the previous chapter. The case studied is a highly complicated product definition system under development in one of the biggest car manufacturers in the U.K. The development method used is evolutionary prototyping approach using object-oriented techniques. Section 4.2 is devoted to a full description of the case study and its findings. It explains the case study in five subsections. First the background of the autoindustry and the company will be presented in order to gain a better understanding of magnitude of the problem and complexity of the environment. Then in subsection 4.2.2 an explanation is offered of mass customisation strategy compared with other strategies in the industry and how the company is planning to move towards this strategy. This subsection suggests characteristics of the business change environment and how it affects the development of a highly complicated information system. This provides the rationale behind the software system and its development method. Information technology projects in general and the product definition system in particular - as the focus of this case study- are the subject of subsection 4.2.3. The subsequent subsection discusses the results and findings of the case study and puts them in the form of an argumentative description about the shortcomings of the evolutionary approach. Finally subsection 4.2.5 summarises the main problems associated with the evolutionary prototyping approach, and two critiques of the principles of evolutionary development which underpin this conceptual practice are developed: the lack of an effective implementation process management system and also the lack of a "support" model for evolutionary development. Finally on the basis of the findings of the case study and the theoretical framework presented in chapter 3, the main argument of the research is formulated in the concluding section of this chapter.

4.1 Introduction

This chapter focuses on the investigation of information systems design and development in a changing business environment. It reports the results of a case study conducted in a large car manufacturer in order to examine the practical adequacy of evolutionary prototyping approach to development. It establishes the main focus of the research problem and links it to the theoretical framework presented in the previous chapter. The objective is to take the research focus from the findings of the case study through the formulation of concepts to their validation and verification.

The case study, employing participant observation of the development environment and in-depth interviews with developers, had an exploratory function in order to gain a better understanding of the key issues related to the evolutionary approach.

4.2 Exploratory case study: Product definition system

This section explains a highly complicated "product definition system" planned in one of the biggest car manufacturers in the U.K. The most important objective of the product definition system is to respond to business change triggered by the product customisation policy adopted by the company. The development method used in development of this system is the evolutionary prototyping approach using object-oriented techniques. The size of the company, volume of information requirements, number of users and complexity of the system, together with the development approach, make it a very good case for assessing the evolutionary approach in developing information-intensive systems.

We will explain the case study in five subsections. First the background of the autoindustry and the company will be presented in order to gain a better understanding of the magnitude of the problem and complexity of the environment. Then in subsection 4.2.2 we compare the mass customisation strategy with other strategies in the industry and discuss how the company is planning to move towards this strategy. We think that it is necessary to discuss the characteristics of the business change environment and how it affects the development of a highly complicated information

system. Section 4.2.3 examines information technology projects and the product definition system as part of a complete new software system which is the focus of this study. The subsequent subsection will discuss the results and findings of the case study and put them in the form of an argumentative description about the shortcomings of the evolutionary approach. Finally subsection 4.2.5 will summarise the main problems associated with this approach.

4.2.1 Background

This subsection establishes the broad context of the case in this chapter. The aim is to provide a better picture of the size of the industry and its associated problems.

Automanufacturing industry

The auto industry is one of the most sophisticated industries in the world linking up different technologies and design skills. Cars and light trucks manufacturers have the largest market share. The world's top 30 manufacturers of cars and light trucks produced more than 44.7 million units with at least 850,000 million dollars of combined corporate revenues just for the year 1992. The market breakdown for 1992 was:

U.S.	33.6%
Japan	34.7%
Western Europe	25.4%
Eastern Europe	2.5%
Other Asia	3.8%

As an example for better comparison of the size of the industry, we can see that U.S. companies built 15.1 million cars and light trucks in America and overseas with combined corporate revenues of $270 billion as compared with 15.6 million cars and combined revenues of $247 billion for Japan in 1992 (Fortune International, 1993).

The UK car industry

There have been enormous changes in the British car industry over the past two decades: the contraction of the industry in terms of the number of people employed in the motor industry, the destination of products from domestic market in Britain to a European market and changes in the technology used to manufacture cars (Financial Times, 1993).

But perhaps the most dramatic change is the extensive use of information technology in the industry. All car producers make use of computer networks to link different parts of the production process. Some people have suggested that technology is responsible for the radical changes which have occurred over the past 25 years. Technology, and specifically information technology was, and still is, seen as the most important component of the regeneration of the UK car industry. Today, the use of computer integrated manufacturing is seen by many as the driving force in the European quest to compete with US and Asian countries in their avowed attempt to dominate the European continent with their products (Computer Weekly, 1992). At the heart of the preoccupation with technology has been the notion of technology driven industrial progress, organised around automation and information technology.

The company

The company under study is one of the world's top 30 manufacturers of cars and light trucks with an annual turnover of 5 billion pounds, car production of around half a million cars a year and the prospect of rising profits in the years to come, in spite of its profits crisis during the recent recession.

The company is a corporate group which is split into two different companies: one for different types of cars and the other specialised in 4-wheel drive vehicles. For the purpose of simplifying references to the two components of the company, henceforth they will be referred in as ***Cars*** and ***4x4.*** Currently *Cars* makes approximately 20,000 cars each week. *4x4* makes 1200 to 1500 cars per week. The net effect of such a production rate is that every 57 seconds, one car must be completed and delivered by assembly lines.

There are two different managerial philosophies and production styles prevailing in *Cars* and *4x4*. Even their approaches to systems and organisational behaviour have also been different. *Cars* was suffering from poor quality and design before this decade, until they decided to apply the Japanese style of management, design and marketing approach to their existing procedures. Traditionally, *4x4* has had a good and profitable market during past decade with one of the best-known marques in its field. They have also established an efficient marketing network around the world, especially in the United States.

4.2.2 Company's corporate policy: toward continuous business change

The company has adopted strategies to change the whole vision of the organisation. They intentionally planned for extraordinary customer satisfaction in order to drive toward continuous improvement strategy, and then from there to targeting for a mass customisation strategy. One particularly useful framework, which describes these strategies and changes in business environment, is the product-process matrix (see figure 4.1).

As shown in figure 4.1, one dimension describes the extent of changes in the demand for new products or services- **what the firm delivers**. These changes may be stable, meaning slow but permanent, or dynamic, meaning quick and unpredictably turbulent. The other dimension describes the extent of changes required in the process and technologies used to produce and deliver the products or services- **how the firm does it**. These processes may also be stable or dynamic. The permutations define four business environments in terms of their change characteristics. Each environment encourages or even demands a particular strategic focus that is most appropriate for success. They are called mass production, continuous improvement, innovation and mass customisation.

The mass production strategy (stable product, stable processes) used to be typical of all large manufacturing and service companies. It exploited the stable conditions prevailing in post war economics to produce large volumes of standard products or services at low cost. A well known example, until recently, would have been General Motors. It is

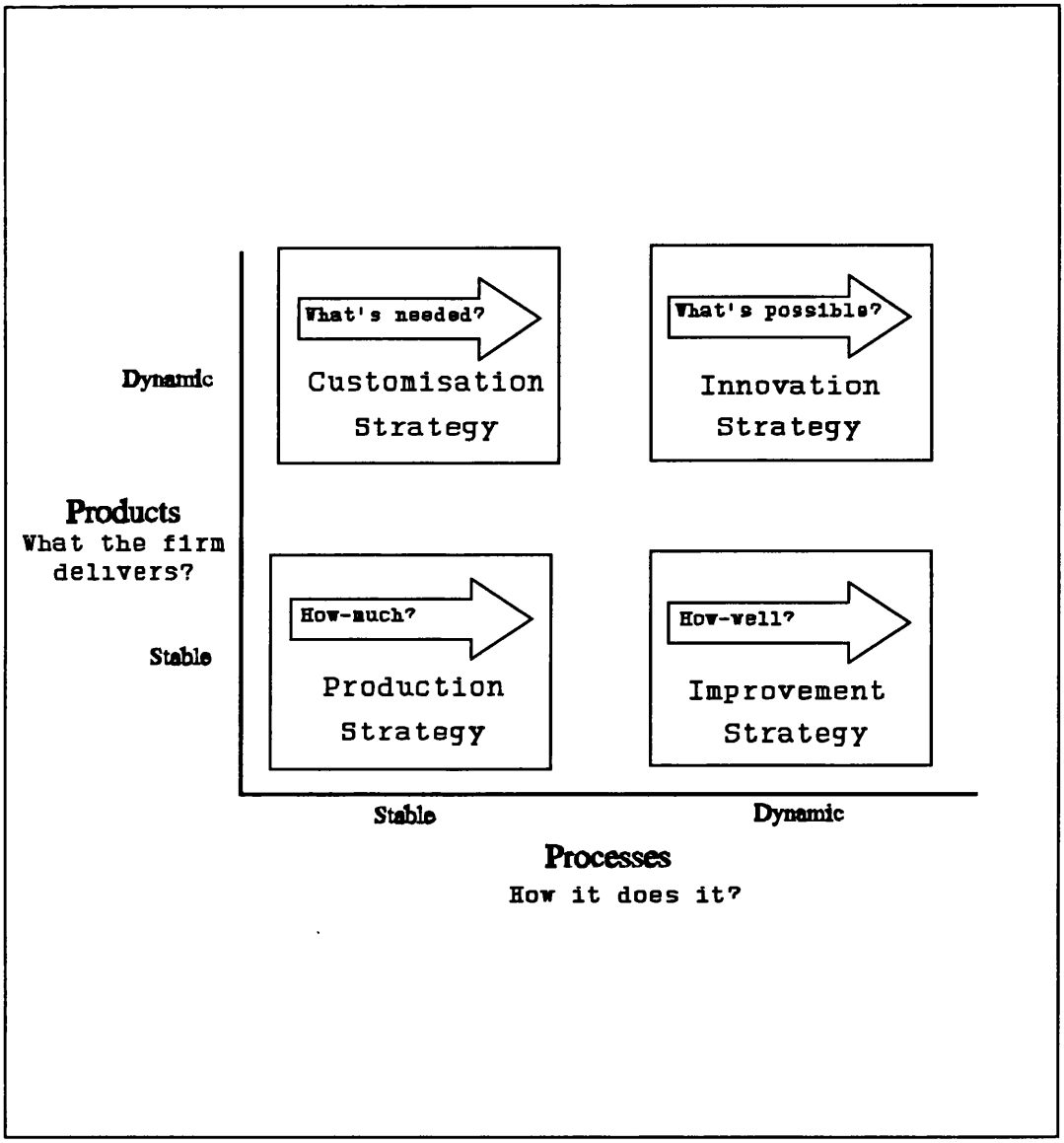


Figure 4.1 Towards business change environment (adopted from Boynton *et al.*, 1993)

hard to imagine any business, other than a natural or protected monopoly, pursuing this strategy today. On the other end of the matrix, the innovation strategy (dynamic products, dynamic processes) is appropriate when the environment is totally dynamic and disruptive, so that both product and process changes are needed continuously. Rapid product change (e.g. introduction of a new hardware system every six month) in information technology companies is the result of rapid movement between mass production and innovation strategies. The continuous improvement strategy (stable products, dynamic processes) has been adopted by most businesses, with varying degrees of success, and especially in car manufacturing companies. Competitive

pressure has forced them to respond to customers' demands for higher quality, lower costs and shorter cycle times by continuously introducing improvements to their operating and management processes (The IT management Programme, 1994).

Today, organisations are moving away from the mass market and are focusing instead on individual customer needs. There is a trend towards mass customisation strategy (dynamic products, stable but flexible processes). In this strategy companies are no longer defined by the products they produce, but by the competencies they posses. Most car manufacturers are moving towards a customisation by assembly strategy that enables customers to select options from a specification list and receive a customised car. Information technology is really crucial in delivering a customisation strategy. It is recognised as a special agent of change and enabler of such a strategy.

Corporate policy

The company has also adopted policies ensuring the success of the customisation strategy. These policies have the most influence in all corporate activities and prospects from organisational culture to personnel management and deploying new technologies. The main pillars of this policy can be expressed in the following terms:

"Job for life"

If the company employee is willing to move and take a new job, where the company will pay for relocation and retraining, an employee can assume that he will have a job with the company for the rest of his life.

"Total Quality Management" (TQM)

TQM principles are based upon the thinking that imperfect output is wasteful to company resources; due to the time it takes to correct imperfections. Therefore, both *Cars* and *4x4* components have a "Do it right the first time" approach to their production management. This policy also covers all the suppliers of the company.

"Just In Time" (JIT)

All suppliers and the company itself are committed to delivering everything perfectly and just in time to warrant the lowest inventory necessary for company and suppliers. Once this is achieved, both company and the suppliers will free up millions of pounds annually, otherwise tied into expensive stock. The JIT approach results in a ½ day of stock at company, and also in ½ day reserve of stock at the suppliers' sites.

The significance of the above approach may be seen more clearly, if one considered that in order to keep a stock of all available options of cars for each of the car models, the company would have to store 1,000,000 cars, costing billions of pounds.

"ExtraOrdinary customer satisfaction"

The JIT policy must not have any side effects on quality and more importantly, on the delivery time of customer's order. The company's mission of extraordinary customer satisfaction comprises the "Efficient System of Distribution" concept (ESD).

In this JIT/ESD mixed policy, the company aims at a time lapse of just 14 days from order of a car to its delivery. This will keep stock down and save money. The key issue here is the responsibility of the whole system for continuously getting better, making no production errors, and reducing costs. Order processing in a 14 day targeted delivery time is an extremely important aspect of company's current strategy and cannot be achieved without the support of the integrated information technology systems.

The other dimension of extraordinary customer satisfaction policy is in marketing planning. The competitive market of the autoindustry demands "a customer driven marketing policy" rather than just focusing on the "quality driven market". The ultimate effect of this approach is **"product customisation"** which it means providing as many options as possible for customers to configure their preferred selection. Product

customisation entails breaking up the tightly integrated networks that form the backbone of a mass production policy and creating a loosely-linked collection of autonomous modules. Each module performs a different task and is perpetually reconfigured in response to customer demands. Automation, typically, is the key to linking these modules so that they can come together quickly and efficiently. Product customisation organisations never know exactly what customers might ask for next. All they can do is strive to be more prepared to meet the next request. To that end, information technology is crucial in product customisation policy. Technology still automates tasks where that makes sense. Certainly, technology must augment people's knowledge and skills, but product customisation requires that technology must also automate the links between modules and ensure that the people and the tools necessary to perform them are brought together instantly. This can be only achieved by information technology. Communication networks, shared databases that let everyone view the customer information simultaneously, computer-integrated manufacturing, workflow software, and tools can automate the links so that a company can summon exactly the right resources to service a customer's unique desires and needs (Boynton, Victor, & Pine, 1993; Pine, Victor, & Boynton, 1993).

Product definition under product customisation policy

In companies with a product customisation strategy, there is no such thing as predefined products. All they have is collection of options and features which can shape a specific product based on the specific customer order. A definition of the (final) product just does not exist. In order to explain the effect of the corporate policy on production planning and technological change, it is necessary to explain the notion of product definition in complex manufacturing companies.

Traditionally, manufacturing has been about parts, and the **Bill Of Material (BOM)** shows the breakdown of parts for each separate product. BOM is a diagram of the components of a product, and reflects which parts go into a sub-assembly, as well as some information about each component, such as the number of a part required for each unit of product, price, supplier, weight, etc. A traditional BOM for each product would look something like the figure 4.2.

In this simple approach toward product definition, we need one BOM for each type of product. But because in a car manufacturing company with extraordinary customer satisfaction policy in terms of product customisation, there are 1,000,000 available products (potential combinations of features), it is impossible to keep 1,000,000 Bills Of Material. So the way in which we resolve this is by defining each product as a combination of features where each feature can have its own Bill Of Material. Through this procedure, we can reduce the required information for product definition of effective products by the every possible combination. Now we just need to have a BOM for each feature.

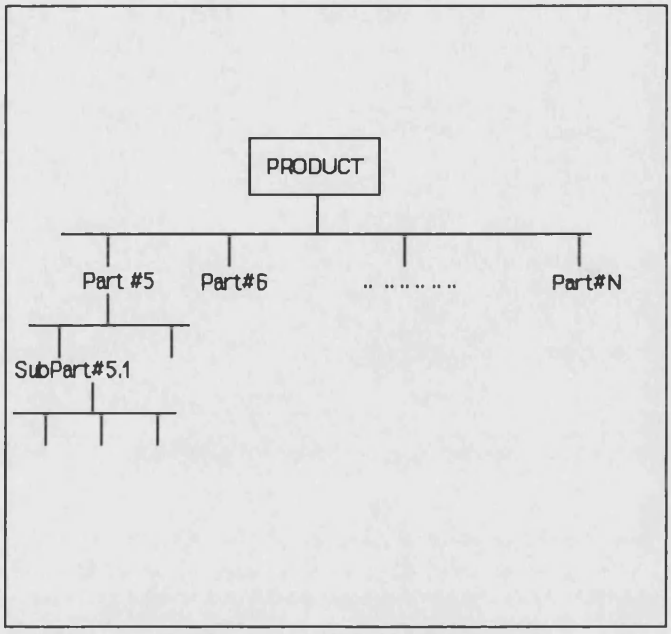


Figure 4.2 Traditional Bill Of Material

Figure 4.3 shows the concept. Here features are subassemblies of which any combination defines a product with a generic name "CAR". Each feature also has its own BOM. A feature for any product with the generic name of "CAR" might be air conditioning, CD radio, red metallic paint, 2.3 litre engine, fuel injection or an electric sunroof. A more realistic diagram of the "New Bill Of Material" configuration should look like the figure 4.4.

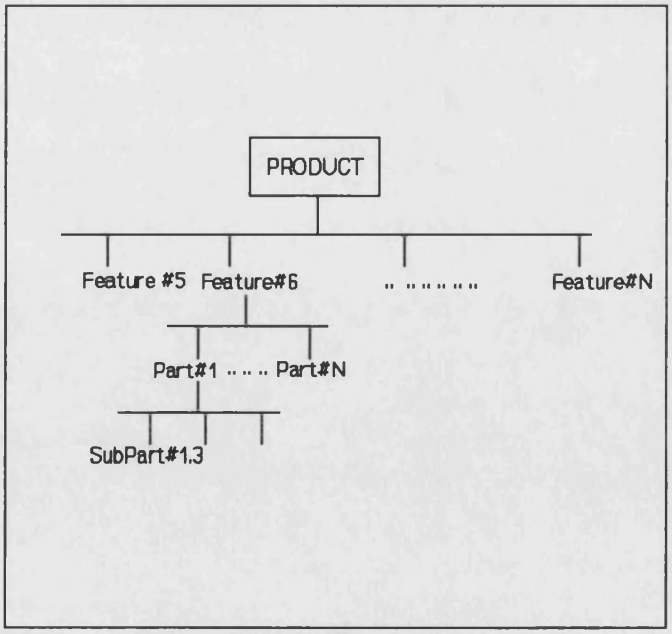


Figure 4.3 New Bill Of Material

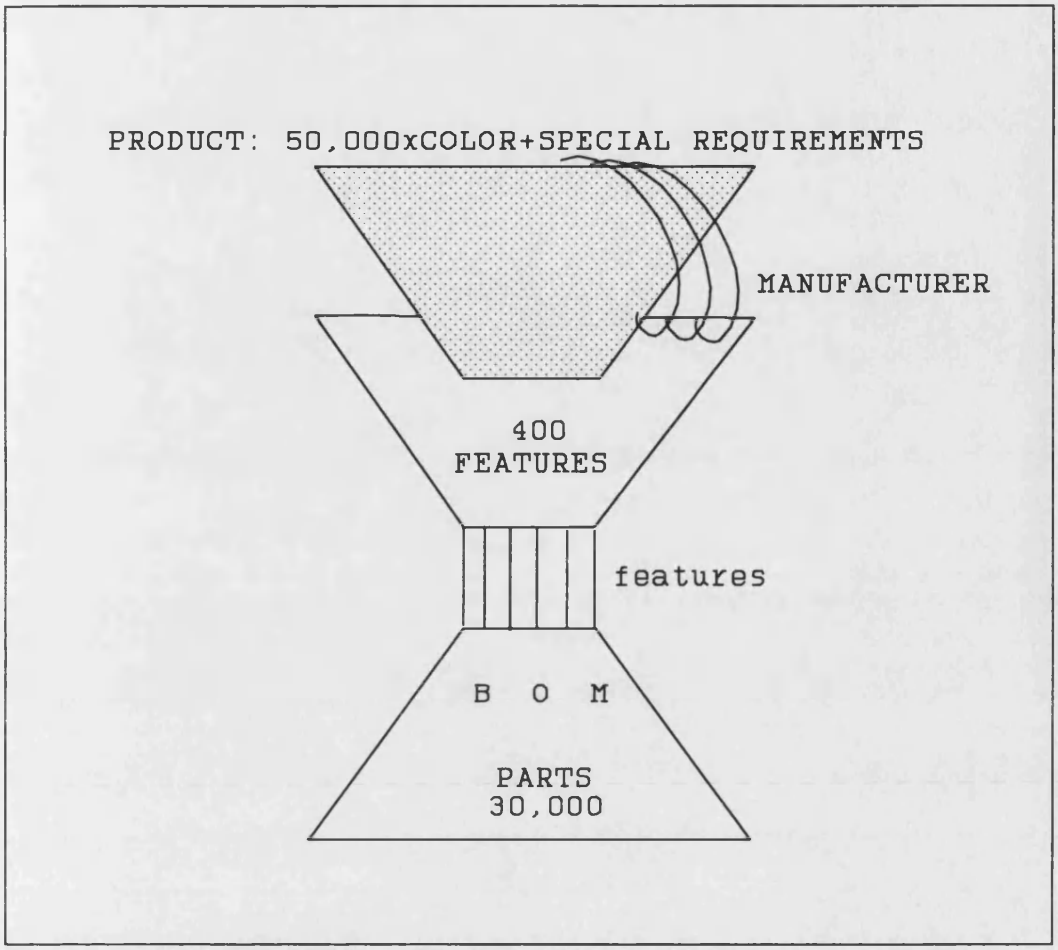


Figure 4.4 New Bill Of Material with combination of features and parts

Production policy

The production policy is one of the most important parts of corporate policy which necessitates supplying better quality products, new options and styles at the right time for the customer. An extraordinary customer satisfaction policy requires:

- more feature options for each product
- acceptable price for each possible combination of features list
- satisfactory default and standard features in catalogues

So what these objectives really require is that the company becomes a product customiser. It means the company has to advance continuously its goals of offering customers a wide range of options and of delivering a made-to-order car within few days. The company must develop much more flexibility in production planning and

operations to become capable of handling a large degree of complexity. The net effect of this policy is the elaboration of an integrated information system for the company as the main part of an automation plan. The information system must have modular capabilities, with the potential of reconfiguration and rapid application development. Flexibility in networks is the other important characteristic of the information systems in a product customisation policy.

4.2.3 Information Technology and production policy

Information technology for mass production policy has quite different characteristics when compared with those appropriate for product customisation policy. In a mass production policy the company needs to have separate infrastructure for each product and service. Each major product has its own dedicated and integrated applications. Under this policy, the company needs to build an efficient information technology system for a horizontal information flow which has been vertically divided according to different products. But an information technology policy driven by a product customisation policy requires a completely different approach. A modular capability with facilities for reconfiguring the information technology services is crucial for an instant response to each requested customisation. Rapid response, vertical information flow, and a flexible network for connection the loosely-connected production modules are the most important aspects of the information technology services in this policy.

Current information technology structure

The company has many development plans in information technology. At present *4x4* has an integrated series of databases on an IBM mainframe, but as a result of major reorganisations and the sale of several major divisions and subdivisions, systems at the other division, *Cars*, have been split off and the whole information technology structure has suffered immensely. Consequently, *Cars* now has over 50 different information technology environments, which the company would like to integrate. Ultimately, all systems for *4x4* and *Cars* will have to be integrated as a management policy for better synergy between the two divisions. At times, it becomes difficult for management in *4x4* to understand completely and to agree with this policy.

Information technology projects

Towards the end of 1989, a comprehensive information technology plan was developed for the following five years. The main purpose of this plan was to develop an integrated information system consisting of a new groupwide Bill Of Material programme and efficient distribution system programme. The Bill Of Material programme covers the full product specification process from brochure model products to each individual part. The second programme is intended to develop a better way of marketing the company products in line with the corporate vision for extraordinary customer satisfaction. This plan consists of different modules (figure 4.5).

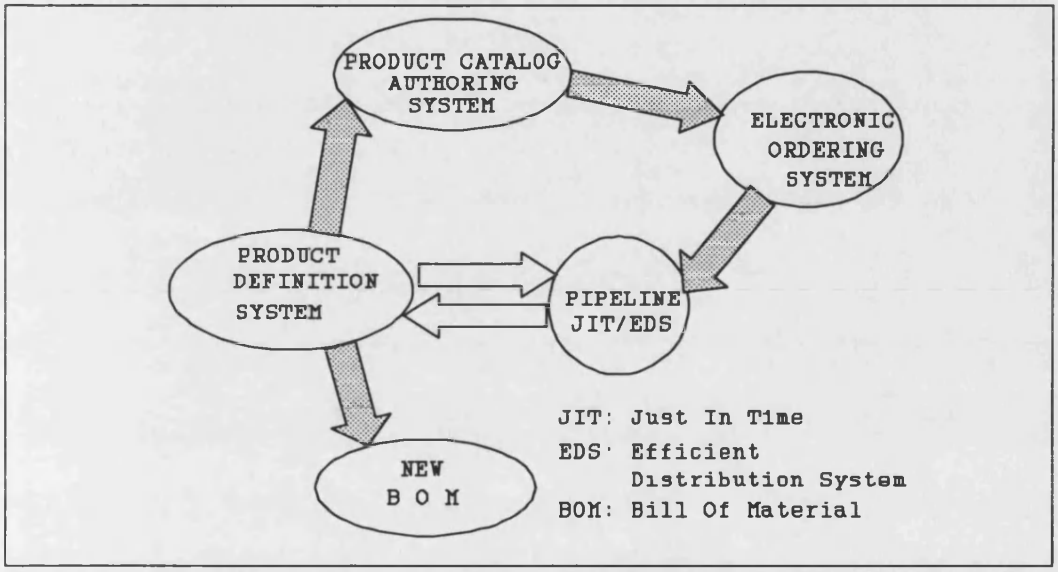


Figure 4.5 IT projects towards achievement of product customisation policy

The main objective behind this system comes from the extraordinary customer satisfaction policy. The following transaction chart (figure 4.6) and its descriptions convey the idea.

As it is shown in the following figure, information technology projects consist of four separate major projects. The product definition system at the centre of the chart has the main role for message passing and connectivity among different components. The four components of new information technology system are:

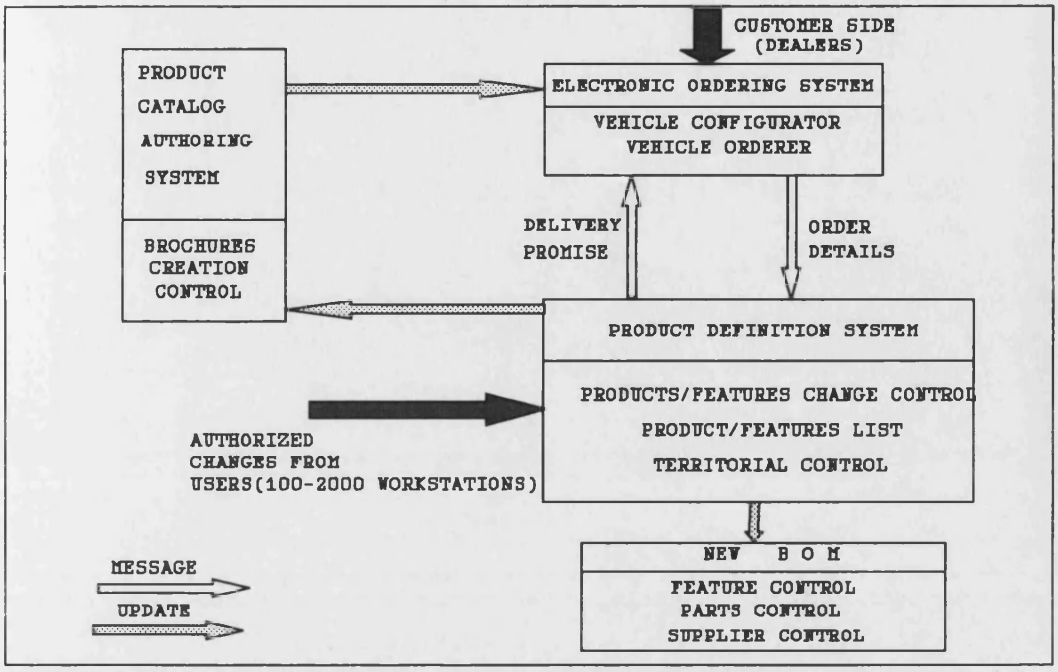


Figure 4.6 Overview of transactions in the new IT system

• **Electronic Ordering System:** This system will hold and validate a customer order by verifying that a combination of features is actually a valid product offering for that particular territory, using information first produced by the product definition system.

• **Product Catalog Authoring System:** This system will facilitate the product configuration for different marketing channels in different countries, based on technical and territorial conditions and features availability in the production definition system. This system will give marketing departments a system to exploit niche markets and customer targeting with minimal engineering cost. Configuration of the feature content of a product in the catalog across the funded life time of the vehicle programme is also handled by this system.

• **New Bill Of Material:** This system will maintain the new concept of Bill of Material based on the parts--feature--product (feature combinations) relationship. The content of each feature and its parts will be supported by this system.

- **Product definition system:** At the heart of the projected total system, this system is responsible for the evolution of any changes in the features configuration of a product. It will provide information for authoring any product catalog after receiving order details and for notifying the new Bill Of Material system. The actual delivery promise is approved by this system.

The next subsection discusses the product definition system as the focus of the case study.

Product definition system

This system is responsible for the evolution of any changes in the features configuration of a product. This system consists of two major modules; full feature specification module and product change control module. The following data are stored and updated by the first module:

- product types, product range (models of cars), feature combination (product), feature lists, feature status (standard, optional or default)

- conditions of feasible feature combination (a rule-base) regarding technical issues

- territorial conditions (like right-hand or left-hand driven regulation in different countries)

- feature availability for marketing requirements and product catalog authoring system

The second module, the product change control module supports lifetime planning for each vehicle, from a broad definition of a new product range to discontinuation of a model and also maintaining any requested change in different levels from introducing new features to minor modification of a feature. Supporting technical, logistical and financial authorization for each requested change forms part of the function of this

system.

The major mission of the product definition system is to enable the evolution of the product specification throughout the life cycle of a vehicle programme by recording and disseminating any requested change. These changes can be based on customer/marketing requirements or product supply support system within the whole group. This system will also support the information about what features and what combinations (by territory) could be required to be offered as a model throughout the life of the vehicle programme. It is expected that the system should be able to deal with over 1,000,000 requests for change per year for improvements of its products.

Time scales and users

The initially estimated time scale for the whole project had the deadline of end of 1994, but became infeasible. The ultimate number of users of the target system was to be approximately 2000 users in the operational environment.

Development approach

The rest of this case study will be focused on the most important module of the information technology project: the Product Definition module. This module has a core role in relation to the other modules and due to its coverage and complexity, the findings on the Product Definition module can be seen as indicative of the overall characteristics of such software development projects.

In phase I of the Product Definition project, a "Clay Model" was developed. The aim of this (throw away) prototype, using Ingres windows 4GL as a requirements prototyping tool, was to show "what the system looks like and does for different people". This approach was used because of the emergence of fast prototyping for obtaining the business support needed for running the project. It was intended that after developing this prototype as an experimental prototype system, developers could plan the complete working system with the experience of an evolutionary prototyping approach. Hence the Clay Model is seen as a very high level requirements prototyping tool. Developers have planned that through this experimental prototyping effort the

major characteristics of the systems requirements (technical and information requirements) will be uncovered.

The prototype tried to convey the idea of product customisation and how the users can define the product changes. The prototype had the capability of representing a feature list of a sample product with the possibility of choosing different features to configure a sample order, without any constraint regarding feature availability or combination feasibility.

The prototype achieved its aim of showing the idea of product customisation and what it will look like at its very preliminary stage. The Clay Model achieved its goal of obtaining the business support for running the Product Definition project. Although this model stimulated the "Business Vision" of product customisation, the working system of the November 1995 specification, which still is not complete, is far from the specifications that the Clay Model was designed to capture. There were many hidden problems in the product customisation concept which have not been addressed by the Clay Model. In that sense the prototype could not achieve its objective as an analysis tool to reveal the extent of complexity in the system.

One of the most important outcomes of the prototype as an experimental prototype was in evaluating the development environment. The prototype tested the speed of relational database technology for the expected speed during search command among millions of different combination sets of features and also its cost for such functionality. As a result the other alternatives were reviewed and an object-oriented approach for database and programming environment was selected. This environment has been found to be cheaper and faster. The development approach was planned in different phases (1 to 3), where each phase consists of different releases (1 to 5) in the form of three to six month projects for each release.

Although the development environment has been changed, the development approach remained committed to evolutionary development: "analyze a bit, design a bit, build and test a bit". So, the system development philosophy is an evolutionary delivery of

functionality with staged business benefits. There are short focused deliverables every 3 to 6 months, with an evolutionary approach to implementation. For the analysis and design tool, "Bachman analyst" was chosen. This tool, which is a modelling tool based on the E-R modelling technique, has been used for documentation of specifications derived from analysis. The transformation of each entity from this model to the concepts of Class/Object in the object-orientation technique is conceptually handled by the development team. One reason for this incompatibility between analysis tool and development environment is rooted in the lack of an efficient and well-known tool (or even mature methodology) for object-oriented analysis and design approach at the time the project began. Developers feel comfortable with this transformation and can understand the concept of Classes/Objects from each entity in the Bachman model. Some characteristics of object-orientation, such as inheritance, are still difficult to convey using this tool.

Roles and responsibilities

The systems development of the project is owned by the "Product Definition" project manager and its implementation will be owned by the department responsible for product supply planning. The User Forum is responsible for guiding requirements and validating the system functionality. The chair is a "user" from the business side, not from systems. At the time of this study the chair was the "product strategy" unit of marketing. Additional users also provide inputs and focus to the development as required.

Each module has a project leader responsible for the day to day running and administration of the project. The company business analyst, together with one person from an outsourced information technology company, is responsible for establishing user requirements and identifying existing and future business practices. Documentation of the analysis phase is also their responsibility. The system functionality is identified jointly by both the User Forum and business analysts. The design, system architecture, software development, test and training are the full responsibility of an outsourced information technology company. A team of five developers, with one person as team leader from the outsourced company, is responsible for the development of each module

according to the requirements specification prepared by the business analysts. Each developed prototype must be approved by the User Forum and then integrated with the previously approved modules. The responsibility for integration and change control in the software system is managed by a team of three highly trained developers from another outsourced company. Feedback from the User Forum to revise the requirements is the responsibility of business analysts.

Focus of study

The study focused on two prototyping efforts. First there was the development of the Clay Model as a high-level requirements prototyping, by interviewing the business analysts and two developers responsible for developing and justifying the prototype. Company documents were also reviewed as a good source of analysis in this study. The second focus was on the evolutionary development project. Most of the findings come from deep interview sessions with the development team and specifically with the team leader, a liaison between developers and business analysts. The following figure shows the main focus of the study.

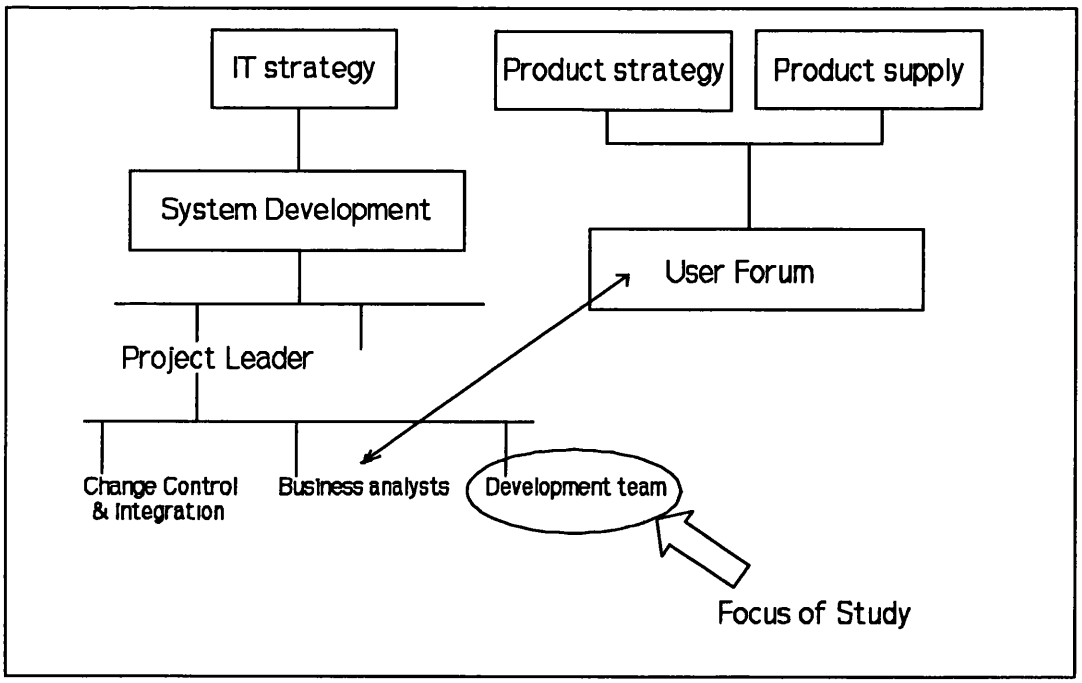


Figure 4.7 The focus of case study

4.2.4 Findings

This section explains the findings of case study and explores some ideas about different aspects of system analysis in terms of specifying user requirements in complex systems. It will analyze the findings of the study in the context of information requirements determination and the associated difficulties in business change environment. It then develops a descriptive argument about each finding in order to achieve a better understanding of the shortcomings of requirements prototyping in general, and the evolutionary approach in particular.

Clay Model as a surface representation tool

Psychology divides the human-made tools into two main categories. One category is called **"surface representation"** and it applies to everything that everybody can understand of the functionality and character of some phenomenon by seeing it, or working with it, like a hammer. So how a hammer works and what we can do with it are clearly understandable from its shape. The hammer is simply a surface representation of its behaviour and capability of hammering. The second category is called **"deep representation"** by which not everyone can understand how to use it or what are its capabilities, like an Automatic Teller Machine. Everyone who has a bank account may use it, but how it works and what are the different capabilities and characters of behaviour are hard to understand by the user, and are intended to be hidden.

Software products, in this dichotomy, are examples of a deep representation of themselves to their ordinary users. Software is a very complicated product based on a formal specification of user requirements. Most of the time, the formal specification cannot represent the deep explanation of the system under study. In such a situation, prototyping as a tool for exploring and specifying user requirements is intended to expose the deep character of user requirements. In other engineering fields, from where the concept of prototyping has come, prototypes represent the depth of ideas about a product in a form that analysts can test the pattern of its behaviour in the real world.

This case study has clarified that prototyping is seen as a "Clay Model" of what a very

limited view of the system might look like. It has not been able to explore the main characteristics of the system under study. Neither the roots of complexity nor the deep representation features of system specification have been addressed by this prototype. In fact the role of the prototype was intended to be a **"Marketing promotion"**, triggering business support of the project. The prototype was not able to reveal the depth of user ideas about product customisation. It seems that this scenario can be generalised to most applications of experimental (or throw away)prototypes in software engineering. The Clay Model was able to provide a benchmark for future development environment and software platform, but due to the complexity of the system under study, it could not reveal the depth of system specifications and its functionality. As a business analyst explained:

> " We inherited the problem of vague understanding of boundaries of the system and with every step we go forward, the incompleteness of the Clay Model and its inability to provide guidance as a feasibility study tool to explore the depth of analysis needed, is becoming clearer."

Matching users' conceptual model with designers'

There is psychological evidence that users develop and maintain their own conceptual model of a system in use (Gittins, Winder, & Bez, 1984). This model develops gradually and enables the user to predict the behaviour of a system on the basis of his limited knowledge (Norman, 1983). There is general agreement that a good prototype should allow the user to develop an accurate conceptual model rapidly and easily. But a serious deficiency of how accurately the users' conceptual model and the designers' one can be matched, has remained unresolved.

In the case studied here, business analysts prepare a conceptual model (in a form of an E-R diagram) by analyzing user requirements and the feedback from the User Forum. This model is then transformed to a prototype specification by the development team (Which also has its own conceptual model) in a very unstructured manner. The result of this process will be a software prototype developed by the development team and then examined by the User Forum. This is the point where different conceptualisations

of the requirements clash and designers hope for a better convergence. But when the system is so big and information intensive and the environment is forever changing, it is very difficult to reconcile these conceptual models with each other. In the case studied, the development team was continuously striving to reconcile differences by undertaking a massive amount of reworking and still they believed that most of the time they had lost the track of the original requirements. The business analysts tried a number of times to communicate with the User Forum by means of their E-R model in order to have one single basis of understanding about the requirements. But the User Forum always found it too complicated, not comprehensive and impractical. The result is that the prototypes are always exercised against a set of criteria which are continuously forming the users' conceptual model. Even each individual in the User Forum has his own concepts about the system and there is no standing platform upon which to unify them. In this situation of conflict, they try to agree upon what the key "factholders" believe. It is found to be extremely difficult to communicate efficiently when no single and understandable platform has been found from which to convey ideas to each other. When the development of a new prototype moves on simultaneously with the development of the organisation itself, a unique communication platform is needed to consider the organisation together with the prototypes. As development team leader explained:

> " We always knew for a large scale system, in order to employ evolutionary development, you need a supporting model to communicate with users and validate the original requirements. Without a support model to maintain the integrity of the whole system, it would be impossible for us to handle the project. But it seems our selection of E-R model to provide the effective support we planned for was not a successful one. (26 July 1993)"

Terms of reference

Fundamental to a system design process is a knowledge of task structure and the user's knowledge and processing limits (Moran, 1981). This requires study not only of the task domain, but also of the user. However, these are fundamentally related since user behaviour can only be examined relative to a particular set of tasks. The first problem

the user has to overcome is to define his system requirements. It has been observed (Malhotra, Thomas, Carroll, & Miller, 1980) that this requires the user to access his memory for sub-goals or solution strategies.

This is extremely difficult as information in the form of user's knowledge can be conveniently accessed only according to the way it is stored. The user cannot anticipate, at the time a concept has to be stored, all the contexts for which it will be useful. The process of remembering can be assisted by irregular memory clues which provoke widely different types of information and bring them into focus. Actual use of a prototype of a system promises to be the best way of generating these clues (Hekmatpour & Ince, 1986). But this process needs to be managed in an organised and systematic way, otherwise with each piece of prototype system an increased amount of information will be produced without our knowing how to categorise it. In the case studied, the business analysts provide a document called "terms of reference" in order to classify related information. It classifies each requirement statements and its reference in the data model under a reference heading. The procedure of assigning and reassigning the requirements statements to these categories is based on the perception by the business analyst of those particular requirements. Each requirements category is then implemented by change control people. Each category demands a particular prototyping effort. The relationship and integration between categories are maintained by the data model. The headings for each category have been shaped through time and sometimes they are difficult for users to understand. Terms like K87 LLFD or BAFC (which stands for Base & Additional Feature Chart system) are common. This system of categorisation of requirements is very inconsistent and hard to free of errors. The business analysts seemed to agree with this criticism, but it works for them. They do not see an insurmountable problem as long as they themselves are involved with the interpretation and maintenance of the categorisation process.

Prototyping user experience

Several important psychological issues relate to the development and use of prototyping. Central to these issues is the value of experience in organisations and the question of how we can design a prototype to highlight the user's ideas about requirements through

access to his experience. Jorgensen (1984) identifies three fundamental aspects of human cognition:

> "The first aspect is experiencing as opposed to being taught factual knowledge and skills. By far the most of the totality of human's skills and knowledge has been acquired by experience. ...Imagine how little sense it would make to learn about gravity in physics if one had not the experience of throwing ball? The second aspect is the ability of humans to perceive and operate at different levels of abstraction... This facilitates understanding of general relationships between concrete matters. The third aspect is the familiarity with those fields of life we have experienced as opposed to those we are only knowledgeable about in terms of descriptions. ... The skill of bicycle riding is a matter of experience. It takes a lot of trying but suddenly you have got the knack..."

Jorgensen (1984) maintains that, although knowledge can be formalized in many areas, humans can operate adequately without having to make this knowledge explicit. He demonstrates this by a simple example:

> "Anyone living in a house with a staircase will know the number of stairs. That is, if asked, an occupant will most likely say "I don't know!" but, nevertheless, inevitably stumble if an extra stair is added on top of the staircase."

The last two examples highlight the difficulties users face when asked about their software requirements. Although users are subconsciously aware of their needs, nevertheless they find it almost impossible to describe them completely. The examples given above illustrate the difficulties a user may experience when trying to visualize a system (Hekmatpour & Ince, 1986). This is the direct effect of how the complexity in systems increases the risk in systems design.

Findings from the case study show that the Clay Model does not address the way that users behave in visualizing and deciding about any requested changes in the product

features. Now in designing the working system, it is very difficult (or to some extent impossible) to formalize all the users' knowledge in the form of definite rules of relationships between different combinations of features. In order to handle these changes the system must be equipped by a very complex rule-based system. As one member of the development team stated:

> "After three years working with the project, now it is clear that the value of the knowledge and experience of product supply people who validate or reject different requested changes in features of a product, was treated very trivially in the first requirements prototyping attempt. They are able to carefully pinpoint any difficulty or mismatch in combination of different features, something which is now revealed to be too difficult to be implemented by computer programs."

This problem is rooted in the inability of requirements prototyping to reveal the depth of user's ideas and to weight them appropriately. The requirements prototyping effort was not able to provide a business picture of the consequences of the total product customisation policy. It has not enough power to reveal the deep structure explication of the user's ideas. At the same time, it is expected that requests for changes in the configuration of a product based on customer needs would increase at least fourfold, once the automated system is completed. This means much more complexity even in comparison to an existing situation of a product supply unit. When users exercise a prototype, they also bring with them all the experiences they already possess. But when it comes to the operation with an ordinary user, the weakness of requirements prototyping in revealing the deep structure of user requirements becomes apparent. Prototypes were used to specify the future product. They failed to communicate in the group of users about what work is done at present.

Change control

Those aspects of the system which are deeply rooted in the structure of the application, tend to be highly connected but to have a low volume of activity, whilst those features which are less dependent on the principles which govern the organisation of the

application area, and hence are more readily subject to change, are less connected but tend to have much higher activity rates.

The subject of change and its control is a very crucial issue in software development and specifically in any evolutionary approach. In a business change environment with greater amounts of information exchange, nothing can remain stable and everything can change at anytime. The following warning appears in every data model produced by the project team:

> "WARNING- This model is being continuously refined. Do not assume that what you are looking at is the latest version. Contact .... for the latest version."

It was really surprising when one of the development team explained:

> "Sometimes I doubt that even business analysts have a clear picture of business activities. During the design briefing sessions, it repeatedly happened that when we ask a question like: How do you want to handle this part? we see an *ad hoc* change in the data model in response to our question!"

It is not easy to separate the deep rooted and invariant aspects of user requirements with the others which are most likely to be subject to change. The focus of evolutionary prototyping brings together the conflicts between what the user wants in an everchanging business environment with what is most amenable to software development techniques. And in this process, it is expected that designer- a person who himself is most affected by the dynamic turbulence of the condition- should resolve this situation.

We need a complete separation of human-oriented design activity and machine-oriented prototypes using a variety of socio-technical tools. Computer formalisms will remain fixed in the realm of signs and symbols. In this form, by using computer prototypes

as the only communication channel between user and analyst, we cannot have a stable basis on which to analyze the principles or basic pattern of the user's behaviour. The descriptive language of analysis, using computer formalisms to demonstrate the user requirements as a prototype, is unable to reveal the indirect inferences of communication acts. Even direct inferences are difficult to draw by relating a change in the content or expressions to its effects.

Semi-formal nature of the systems (Whitley, 1990)

The system under study may range from being simple and sufficiently well-structured to be clearly defined and understood, to one which has an uncertain nature and a complex informality. In the real world, formal systems make only a small contribution in governing organisations. It is not acceptable (or even hardly possible) to formalize the whole spectrum of actions in an organisation. If anyone should attempt this, in most cases the informal groups within the organisation will join together to establish their informal procedures beyond the formal system. This is an example of **"organisational resistance"** against the radical innovations held to be against established interests.

The major difficulty for a systems analyst is to reveal the interrelationship between the formal and informal parts of a system under study. The semi-formal character of systems cannot be sustained by rules alone. This is the glue which binds together the structure of rules of formal parts to the structure of informal norms. The system under study is comprised of organisational behaviour, of which rule and formality constitute the explicit parts and without informal assumptions, goals and cooperations, they have no meaning to hold together those institutions.

The ability to develop a sustainable information system depends upon being able to determine the boundaries between the formal and informal parts of systems. How the semi-formal system links these parts together and how the responsibility can take into account in different courses of action are important issues must be resolved in the process of development. The implementation process needs to provide settings for the product's role and user's role for the future. The result of an observation by the

researcher from an evaluation process of a small part of the developed system showed that during the evaluation of prototypes, each individual who holds an experience continues to use it and change it as well. An evolutionary prototype implementation process needs to be viewed as an opening of the learning process. When this issue was discussed with the development team, they were not ready to admit the problem and presumed that these issues could be addressed during the final tuning or maintenance of the system, as a post-implementation problems and not during the implementation process.

But this could be seen as promising solutions by **"gold plating"** software based on over-formalized solutions, instead of adapting to the reality of organisational behaviour.

4.2.5 Summary of findings

We have examined in depth in this case the shortcomings of requirements prototyping and evolutionary development. The findings of this exploratory case study have shed more light on the problems of the evolutionary development approach in practice. Assessing the effectiveness of evolutionary prototyping in information systems development, we can summarise the important findings of the case study as follows:

- Clay Model as a surface representation tool

In complex systems, requirements prototyping and Clay Model had not the capability to go deep enough into requirements features and their complexity.

- Matching users' conceptual model with designers'

For large scale system development in the evolutionary approach, we need a supporting model to maintain the integrity of the prototype systems. The E-R modelling could not provide the required communication platform between users, analysts and developers.

- Terms of reference

Prototyping generates perspectives which can provoke widely different types

of information requirements and bring them into focus. Where a large scale system is concerned, we need to maintain the overall picture of evolving requirements through a categorisation of requirements which is understandable by users and developers and also supportive of different interpretations.

- Prototyping user experience

Prototypes are continuously trying to specify the requirements for the future system. They cannot effectively communicate to users what work is done at present. The present situation is a combination of organisational rules and norms which needs to be uncovered and understood before starting the implementation of any software system.

- Change control

A prototype, as a machine-oriented tool, does not have the ability to separate the deep rooted and invariant aspects of information requirements from those which are most amenable to software technology. We need socio-technical tools in support of human-oriented design activity to deal with change in user requirements, otherwise designers will be continuously involved in a series of *ad hoc* changes in requirements specifications without knowing what is the next step.

- Semi-formal nature of the systems

An implementation process management system is required to adapt the new system to the reality of organisational behaviour, otherwise the social nature of organisational needs might be lost under the shadow of over-formalised systems, which can result in the failure of the information system.

A full study of the above problems has led to the identification of the following crucial difficulties in evolutionary development as two *lacunae* in the approach:

1) The risk of *ad-hocracy* is always to the fore when using the prototyping approach. In the prototyping approach with a cooperative analysis and design approach to development, developers must empower the users and support their ever-changing requirements. Therefore there is a need for planning the whole process of development, but without a blueprint of how to manage the process, the risk in development is high. The act of investigation changes what the users want, while minor assumptions can become major obstacles. The pendulum of new requests for change swings back after each step forward of the prototype. Developers need a standing platform as a basis for adapting the prototype system to the reality of organisational behaviour.

2) The lack of a support model as a frame of reference for analysis and design during evolutionary development is another crucial shortcoming of the approach. What developers need is a dynamic model-driven approach to development which can model the entire business and its underlying deep structure, not just the series of snapshots of the business which prototypes generally offer. They need a model with the ability to provide a consistent schema and to trace back and validate each prototype against its original user requirements, when evolutionary development approach is adopted. A modelling technique is required to develop schemata of underlying business tasks to support evolutionary development with the ability of **schema evolution**. As the system evolves, new views of the requirements are added by restricting or extending existing data models (or requirements descriptions), and new prototypes on these views are generated using existing prototype modules. Therefore, we need a modelling technique which offers facilities for expansion or modification of existing information requirements schemata. We need to be able systematically to manage this evolution of schema changes and handle the change control procedure through an effective requirements categorisation technique. Such a tight coupling between prototype and schema offers considerably more scope for schema evolution through the extension and refinement of existing

information requirements schemata and the reuse of prototypes. If the model can explicitly address invariant features and the underlying business tasks of the system, this will permit the semantics of schema evolution to be rigidly defined and validated. The case studied here showed the inability of E-R models to support the above needs.

Because of these weaknesses in evolutionary development some researchers are advising the use of this method only for small to medium size systems (Angell & Smithson, 1991; Hekmatpour & Ince, 1986; Land, 1982), where the scope of the problem, the number of users and the list of goals and expectations are reasonably manageable. Regarding the complexity of business change environment, what developers need is a strategy that points the way: a model which can provide some guidelines as to where to start and where to go next, given the context of organisational goals.

This research proposes a new theoretical framework in response to the shortcomings of the evolutionary prototyping. The planned change model can provide a suitable strategy for managing the evolutionary process. The iterative sequences of unfreezing, moving and refreezing align the development project towards the agreed objectives and protect it from *ad hoc* design decisions. The semantic analysis technique provides the frame of reference required during prototype development and evaluation. It has the capability of clustering requirements into subject areas, hence reducing the complexity of requirements analysis and of developing modular prototypes in the moving stage of the planned change model. Because of its support for schema evolution during the unfreezing stage, developers and users can have access to understandable conceptual models agreed upon before developing each prototype. Then in the refreezing stage of planned change model, they can portray each prototype faithfully to its original conceptual model, incorporating the semantics articulated in requirements.

The findings of the exploratory case study highlights the pitfalls of the evolutionary development in practice and directs the support of the proposed theoretical framework in abundance of those deficiencies. This will guide the research to the development stage of its research approach in proposing a new development method for evolutionary

prototyping approach, the subject of the next chapter.

4.3 Conclusion

This chapter offered an opportunity to examine, in an argumentative/subjective mode, the implementation process of evolving information systems. The most important deficiencies in the evolutionary approach to large scale information systems development projects were identified and argued against the proposed theoretical framework. There were two major deficiencies:

- no effective implementation process management model whereas the proposed framework offers one

- no support model for determining information requirements whereas the proposed framework offers semantic analysis

It is important to reiterate that there is no golden path for specifying information requirements and for generating information system specifications automatically. However, when substantial organisational change through development of information systems is expected, it is generally suggested that prototyping is an appropriate approach (Alter & Ginzberg, 1978) since in an organisational context, the prototype model can provide information on a wider range of issues for any envisaged software system. The case presented in this chapter discusses the potential pitfalls of the evolutionary approach in the context of information intensive systems. In the following chapters a re-engineering of this approach will be introduced in the same context. The aim is to architect a new perspective on evolutionary development by demonstrating that organisational change relating to information systems development can be supported by an appropriate support model, where the model itself is rooted in an essentially subjective view of the world. While business survives and prospers in the increasingly turbulent environment, we need information systems development methods to support organisations in their evolutionary journey. Information systems development must be prepared to abandon the relative comfort of its traditional role of application

development. It must set out to experience something different, where the main role is that of semantic broker and conceptual training coach. The rest of this research will focus on a response to that inquiry.

CHAPTER 5

A method for evolutionary development

Chapter overview

This chapter offers the proposed method of the research. Before launching into an explanation of how the proposed method works, various reviews are undertaken. Having explained the difficulties of the evolutionary development approach in section 5.2, the theoretical pillars of concepts in the proposed information system development method are reviewed in following section. Section 5.4 introduces the overview of the proposed approach in three main levels: organisational, conceptual and technical levels. It also shows a combination of top-down and bottom-up cycles in the proposed method. Section 5.5 discusses in detail the new perspective to evolutionary development using semantic analysis and prototyping techniques within the control of a planned change model. Based on the new perspective on evolutionary development presented in the section 5.5, a new development method is proposed in section 5.6 to formalise the stages and to merge them into a coherent whole. Finally section 5.7 will conclude the chapter and summarise the main features.

5.1 Introduction

Prototyping has been in common use as a technique for software development projects for some time. It is primarily a requirements discovery technique, used to help determine the application functionality, data structure and control characteristics of a system (Connell & Shafer, 1995). Requirement specifications are explored through experimental development, demonstration, refinement and iteration. Using prototyping technique, evolutionary development was differentiated from *hacking* (developing conventional software programs without benefit of formal requirements and design specification), and *prespecification* (presupposing all detailed requirements and design specifications before developing any software). Although there are now few arguments

about the validity of the evolutionary development approach, there are still few published descriptions of exactly what it means or how to do it in complex, large-scale systems. We explained the evolutionary approach and its deficiencies in previous chapters, and proposed a set of new theories and techniques to be able to offer a new perspective on evolutionary development for complex, large-scale information systems in business change environment. This chapter will distil all arguments into a coherent formalism of a new method for dynamic information systems development. Instead of tacking a thin veneer of prototyping onto the tail end of semantic analysis technique, we take just the opposite approach. The proven technique of prototyping and the planned change model along with the semantic agent-based formalism serve as the framework into which the new development method are placed.

In proposing a new perspective on evolutionary development, the original goal of evolutionary development is unchanged: accurately reflecting user feedback while evolving a developing prototype towards high-quality maintainable system that meets the users' needs. Only the vehicle used to traverse the path is new. The structured, procedural, hierarchical, function-oriented development tools and modelling approaches proposed in different approaches have been replaced by semantic-oriented, model-driven, change resilient, subject-clustered techniques and modelling approaches. In so doing, improvements are made, and provide dynamic requirements modelling techniques that:

- reflect more accurately the socially constructed real world;
- provide an organised way to tackle the problems of uncertainty and risk in developing information systems;
- reduce long-term system costs by narrowing the focus of each prototyping effort to a specific subject cluster;
- and probe the semantic sensitivity of each subject area examined, in order to prototype the subsections of a large system.

In using the proposed approach, developers have a choice: the conservative drudgery of total prespecification with customer sign-off before implementation, the joyous but dangerous practice of prototyping without specifications, or a concurrent approach

undertaking both the requirements and implementation specifications at the same time. The concurrent approach is more feasible in a business change environment. This is the researcher's belief that semantic analysis as a form of explanatory system performed concurrently with an iterative evolutionary system and applied in three-phased change model is not just a theory - it really works! This chapter will explain how.

5.2 Review of the research problem

The focus of this research was defined as evolutionary development approach to information systems development in business change environments. We discussed evolutionary development as a subset of the prototyping approach, in the sense that systems are designed to be changed. However evolutionary systems evolve in use and not in experimentation. In the use environment, uncertainty is the result of the turbulence and dynamism of the environment, so development can benefit from prototyping which can address explicitly the problems of uncertainty and change in requirements determination. The main thrusts in the evolutionary approach are a stable strategy about the problem boundaries and pinpointing the target system yet denying the feasibility of any final specification (Angell & Smithson, 1991).

The exploratory study conducted at a large car manufacturer company found two important shortcomings in the approach: the risk of *ad-hocracy* when using the prototyping approach and the lack of a support model as a frame of reference for analysis and design. There is a need for planning the whole process of development otherwise the risk of *ad-hoc* activities will be great.

In short, the problem addressed in this research was how to support the evolutionary development approach with a model to cluster effectively the requirements for better determination, to identify viable requirement candidates for prototyping and to trace back evaluation of prototypes using accurate information. The process of developing evolutionary systems and managing development were two further difficulties addressed earlier.

In order to enable an evolutionary approach to developing large scale systems in a business change environment the research needs to focus on development of a complementary method that enables designers to:

1) manage the development process of large systems which can cope both with existing changes in design environment and changes induced to the environment by the development of the new system.

2) support evolutionary system with a model which is relatively stable and risk sensitive and has the facilities of requirements clustering and requirements traceability.

5.3 Overview of the proposed theoretical framework

We have suggested a theoretical framework based on planned organisational change theory and semiotic theory in response to the two objectives mentioned above. The aim is to propose a new method for evolutionary information systems development. In chapter 3, we proposed that semiotic theory and planned organisational change theory can be employed to constitute a new theoretical framework for evolutionary information systems development. The proposed framework provides us with the semantic analysis technique as analytical tool, prototyping as implementation tool and a three-phased change process model - unfreezing, moving and refreezing - as a model for management the implementation process. Semantic analysis will sustain the unfreezing and refreezing stage of development, while technical prototyping will support the moving stage. The semantic analysis technique and the planned organisational change model were considered to be in harmony with each other when employed in an evolutionary development environment.

It was also argued that semantic analysis through its formalism - sematic agent-based modelling - gives us a conceptual model of the business in the form of invariant patterns of behaviour of responsible agents. The model is subjected to rigorous semantic constraints which ensure no ambiguity exists and it seems to remain consistent

throughout the development process. Semantic analysis was also held to be equipped with an important complexity reducing element - the subject area clustering characteristic - which makes the technique easily applicable in large scale and complex business problems. The semantic collection and grouping of requirements with a consistent ontological dependency pose a significant challenge to the effective identification of requirements to prototype. The semantic constraints require that different responsible agents with different cognitive styles can recognise themselves in a unified frame of reference which will serve as a consistent perceptual filter through which all agents can interpret organisational activities and information requirements. By achieving a unified frame of reference in this manner, the results of semantic analysis in the form of its graphical representation - the ontology chart - give the users the opportunity to view original requirements during prototype presentation, offering requirements traceability and a way of evaluating prototypes. The above characteristics of semantic analysis, when embedded into the three-phased model of planned organisational change process, seem to respond to the shortcomings of evolutionary development approach. The amalgamation of both theories with the evolutionary prototyping concept provides the ingredients necessary to form a new development method.

5.4 Overview of a new method for evolutionary development

The aim of this chapter is to propose a new method for evolutionary development. The method distinguishes three major levels of abstraction for an information system development process (Albadvi, 1995a): the organisational level, the conceptual level and the technical level (figure 5.1). At the organisational level, it is possible to create incrementally an enterprise information model. This model of semantically related subject areas results from the natural aggregation of ontology charts created at the conceptual level. By using the semantic analysis technique to determine more readily the changes in information requirements, the enterprise information model contains the picture of the whole business regardless of the current development project. The new method prescribes the use of semantic analysis from the perspective of semiotic theory in order to model the enterprise information structure and maintain continuously the

model. This model assists in the discovery of deep structure information requirements and in clustering subsections of a large system into a set of interrelated subject areas. At the conceptual level, the new method proposes a semantic agent-based modelling formalism focusing on semantics and ontological dependencies for each subject area. The modelling formalism focuses on semantic categories of signs and signifiers. At this level, a catalogue of semantic schemata of ontologically dependent affordances is constructed. This catalogue forms a growing pool of ontologically traceable patterns of behaviour of responsible agents within the workplace. It forms a basis from which to evaluate prototypes. It also represents a powerful mechanism for tracing back each prototype to its underlying information requirements during the evolution of new software system. The underlying information requirements represented in the form of ontologically dependent affordances (as a required pattern of behaviour expressed in requirements analysis) can be mapped to the specification of prototype systems developed at the technical level. This should lead to yet higher level of requirements traceability and prototype reuse. The third, technical, level is involved in development of prototypes for each subject area modelled at the conceptual level.

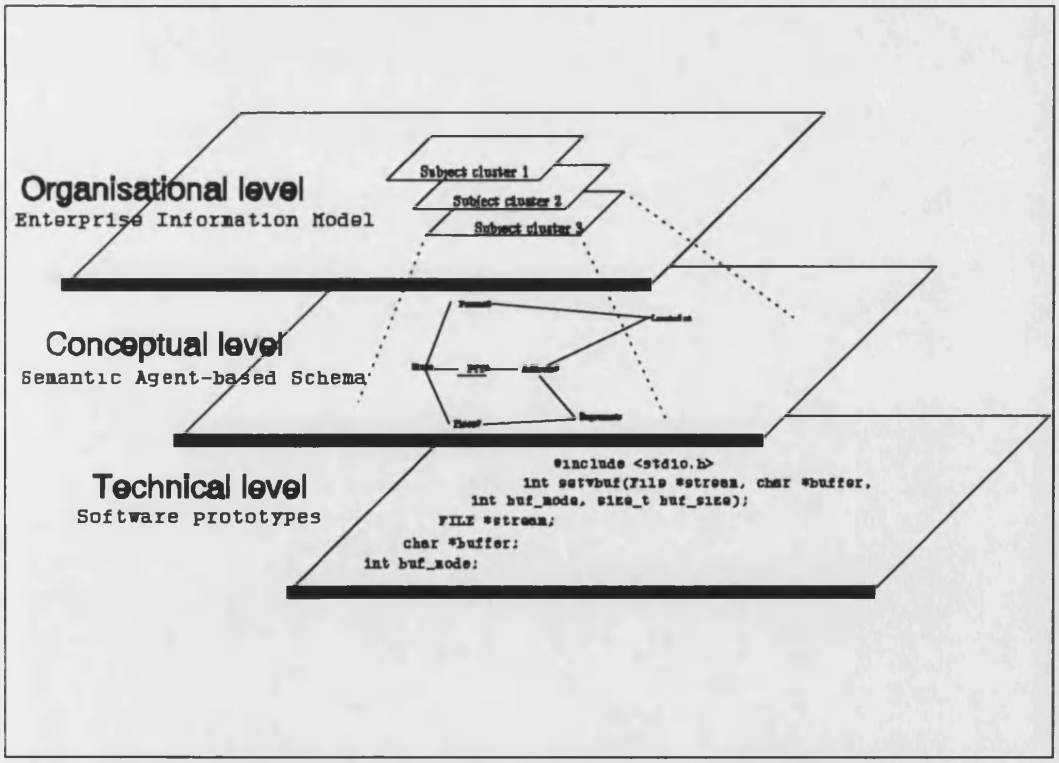


Figure 5.1 Three levels of abstraction in the proposed method

The development process using the planned change model provides both a top-down and a bottom-up cycle to development on three levels. In a **top-down** cycle, semantic analysis is employed at the organisational level to investigate the existing norms and working routines within the work place. It initiates the unfreezing stage of change by focusing on semantic primitives of information requirements. The results of semantic analysis will be clustered naturally and represented in the form of ontology charts at the conceptual level. The conceptual level also contributes to a feed-forward process, the unfreezing stage, by highlighting the semantic ambiguities and establishing a felt need for change. The third level in a top-down cycle, the technical level, involves making the actual change possible. It gears the moving stage into the change process. It involves development of actual software prototypes which give users the opportunity to examine the changes expected in the new information system. Evaluation of prototypes at the technical level initiates a **bottom-up** cycle in the development process. The aim of this feedback process is to portray each prototype at the technical level in the light of its original information requirements, represented in the ontology chart at conceptual level. The bottom-up cycle becomes the refreezing stage of the change process. The aim is to ensure that the new behaviours can become the operating norms at work, without any ambiguity. The refreezing process, through continuously evaluating prototypes and mapping them back to the semantics in the ontology charts, entails the integration of new attitudes and behaviours into persisting patterns and relationships. It is in this bottom-up cycle that the responsibility structure and norm configuration of the new information system need to be clearly understood and established. The cycles of feedforward and feedback processes may result in the approval of the prototype system at the technical level and its corresponding ontology chart at the conceptual level for each subject area. Finally at the conclusion of the bottom-up cycle, the ontology charts for each subject area are integrated into a complete picture of the business at the organisational level. This final stage at each cycle of development requires institutionalising new behavioral patterns by making them organisational norms. This stage, through accrual of approvals of the change process, provides us with an incremental creation of an enterprise information model for the whole business. This concludes the refreezing stage of the change process.

The proposed perspective intrinsically favours strong participation. It favours a systems development process that stimulates consensus participation and induces change in systems of interpretation. Therefore, the assumptions of this perspective do not address the notion of power. It is necessary to mention here that approval and integration of ontology charts at the organisational level requires a diagnostic analysis of power within the organisational structure. We recognise the possible criticism of "naive consensus" (Habermas, 1984) in the proposed perspective, but regard the detailed addressing of the power issue outside the scope of our present research.

5.5 A new perspective to evolutionary development

In theory, the activities of analyzing the problem and of creating a solution are clearly separated. However, in practice there is a gradual move from one activity to another, with a solution forming before all details of the analysis are complete. This is exactly the situation in the proposed method.

As in figure 5.2, the proposed model consists of three cycles: two inner cycles incorporated within an outer cycle. Two inner cycles are the **vision cycle** and the **action cycle**. Both inner cycles have their own cyclic pattern while both are part of an outer cycle called the **fusion cycle** (Backhouse & Albadvi, 1995). The new method is based on verifying the vision-action-fusion cycles shown in figure 5.2. When the feedback stays within the current cycle, it is represented by the inner loop; when the feedback is to subsequent cycle, it is represented by the outer loop. The proposed development method sees information systems development as a continuous process and the three cycles of the development

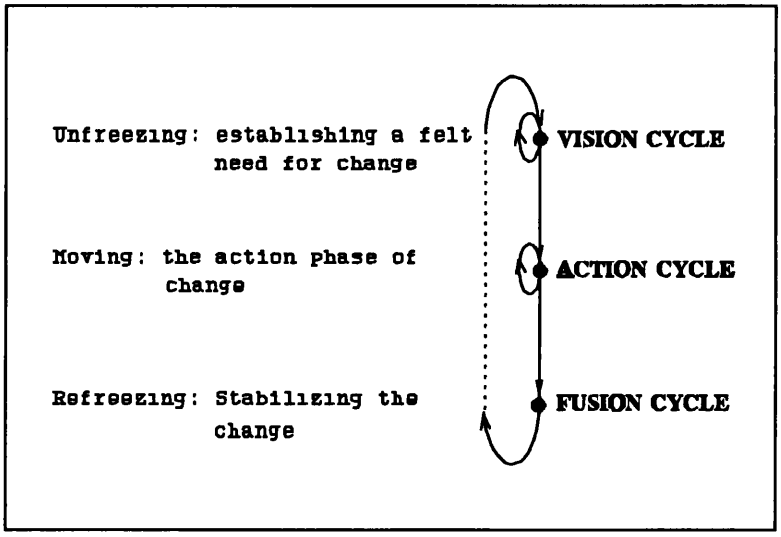


Figure 5.2 Three cycles of the proposed development method

process have to be followed repeatedly.

5.5.1 Vision cycle

In the vision cycle, the analyst deals with problem definition, the elicitation of user requirements. Realising what are the information needs and revealing the nature of the problem through a process of semantic analysis is achieved at this cycle. Applying the semantic agent-based formalism makes it possible to develop an ontology chart of the focal system under study. The ontology chart acts as an explanatory system in order to provide explanations about the semantics of requirements and norm relationships within the workplace. This is indeed a prototype of an explanatory system which remains to be examined and validated by user groups. The long-term product of this process is a completed explanatory system - a rich picture of the whole organisational behaviour and norms. Every explanatory system prototype is involved in a representation of the problem and is concerned with stating the meanings and properties of the information requirements precisely and unambiguously.

Every representation of the problem needs to be validated: the process of ensuring that the problem has been clearly understood and modelled. This is the reason that this cycle is called vision cycle. In this cycle, a process of prototype-exercise-validate and revise the explanatory system is planned (see figure 5.3). The explanatory prototype system provides a base model of the organisational behaviour to support agreement about the semantics of requirements. The vision cycle initiates the unfreezing stage of the change process. It provides a series

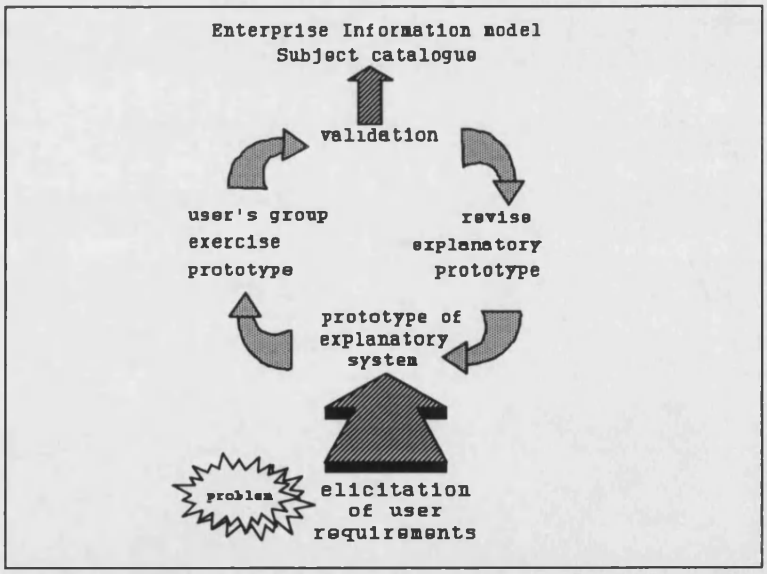


Figure 5.3 The vision cycle of the proposed development method

of explanatory prototypes as ontology charts in order to resolve semantic ambiguities in requirement statements.

The aim of the vision cycle is to provide a unified view of each subject area shared by users. We apply the semantic analysis technique in this cycle and the result is a set of explanatory prototype systems that need to be placed in the requirements specifications. To reach the level of requirements specification, it is necessary that the analyst plans for an approved explanatory system in the form of a complete and refined ontology chart for each subject area. A process of prototype-exercise-validate-revise the explanatory system encourages all responsible agents to seek for consensus where there exist semantic disagreements before committing to development of any software prototype system.

The explanatory prototype system provides a platform for more communication and for the mutual understanding needed for the change process. The different cognitive styles of different people (including users, analysts, developers) can merge together through an explanatory system. This will establish semantic equivalence and bridge the semantic gaps among user groups, so they can all understand the system under development. The vision cycle not only demands a responsible user involvement, but also requires that users form shared and realistic understandings and expectations about the system. Exercising the explanatory system and discussing semantic ambiguities among user groups provides a medium for conceptual training by facilitating the negotiation of meanings. Every review of ontology charts provides conceptual training towards assuring that users indeed hold a unified frame of reference about the system being developed. More time spent in the vision cycle in order to mould a unique frame of reference would lead to less time, effort and cost in software prototype iterations and in conventional tuning and maintenance efforts after prototype developments.

5.5.2 Action cycle

While the user exercises the explanatory system of each subject area in vision cycle, the validated parts of ontology charts of the focal system can be added to the enterprise information model. The enterprise information model contains the semantic schema of

the whole organisational behaviours and norms clustered by subject area. Each iteration of the vision cycle results in more detailed information about requirements specifications, which as a whole creates the enterprise information model of the organisation under study. The enterprise information model is a baseline of semantics for organisational activities. The enterprise information model is the ultimate frame of reference in organisational norms and rules. As a result, incremental modelling of enterprise information structure is possible. This possibility is achieved through the modifiability and extendability of the semantic agent-based modelling formalism. The enterprise information model is clustered into the subject areas. The subject catalogue is a directory of all subject areas modelled as part of the enterprise information model. This catalogue of clustered subjects indicates a name for each subject area which conveys its contents. It also represents the relationships between areas through their common affordances. Each subject area consists of one or more ontology charts. The consistency of the subject catalogue is maintained automatically by the stringency of semantic constraints applied in ontology charting. It is not possible to assign an unrelated affordance to a subject area, as it will be prevented ontologically by existing affordances in that subject area.

At this stage those parts of enterprise information model suitable for automation in the objective system are identified. We call this cycle the **action** cycle. The relevant formal fraction of each subject area in the enterprise information model will be prepared for software prototyping. The level of automation required for each subject area must be defined by user groups and examined through software prototyping. In this process, software prototypes will be developed on the basis of the semantics of user requirements declared in the form of ontology charts. Ontology charts are already clustered into subject areas, so the prototype development and change control activities can be managed separately for each subject area and we may be sure that the semantic integrity between prototypes is effectively maintained at the enterprise level. The explanatory systems in the vision cycle which have required more iterations to resolve ambiguities and to offer a unified ontology chart are the riskier ones. They also indicate a higher level of risk in development of the actual software prototypes in the action cycle. The risk of unacceptability of the software prototype increases when its

subject areas seem to be sensitive to different users' cognitive styles. This provides us with a viable mechanism to identify candidate subject areas with priority for prototyping.

Evolutionary prototypes developed in the action cycle need to be validated. In this cycle, through a cycle of prototype - exercise - validate - revise, it will be possible to develop a solution; a representation of the system specifications. As shown in figure 5.4, while users exercise the evolutionary software system, the specification of the validated parts of the prototype system can be amended in line with the system specifications. Each iteration of the action cycle progresses the development of the target system. But the essence of evolutionary development denies the feasibility of any final target. Each exercise of software prototypes by users provides a feedback within the action cycle for an improved revision of the software prototype system. It also provides a feedback to the vision cycle requiring changes at the conceptual level. Every revision of the software prototype also provides a new outlook on the validity of the explanatory prototype system developed in the vision cycle. Actual software prototypes expand the visionary focus of user groups and may offer them a new depth on their vision of the business. This feedback to the vision cycle will foster the characteristics of the new system within the understanding of the users. It initiates the refreezing cycle of development which we called the fusion cycle.

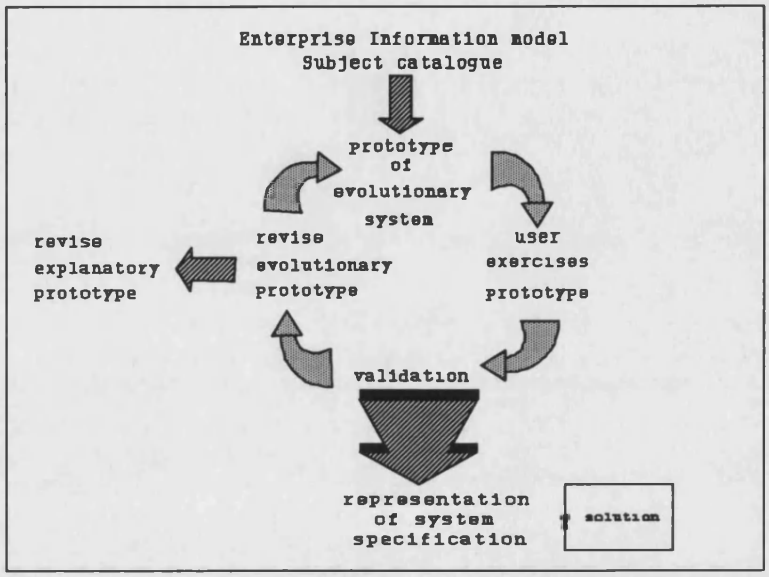


Figure 5.4 The action cycle of the proposed development method

5.5.3 Fusion cycle

In the proposed method there are two concurrent processes. One is the explanatory prototyping process during the vision cycle and the other is the evolutionary software prototyping process during the action cycle. Through the bridge from the vision cycle to the action cycle, the analyst can develop the enterprise information model. This is a feedforward link between the results of requirement analysis and the implementation of software prototypes. The feedforward process stimulates the moving stage of the change process. The development of software prototypes and multiple revisions in the evolutionary system tests the action stage of the change process. Every movement toward a new change condition provides some feedback to the existing patterns of behaviour expressed and captured during the vision cycle (see figure 5.5).

Every revision of the evolutionary prototype provides some more insight into the changes required in the explanatory system. Since the semantic schema, the support model for the explanatory system, of each subject area and the software prototype, representing the evolutionary system of that subject area, are organically related it is possible to trace back changes from the evolutionary system to the original requirements in the explanatory system. Through this feedback process, it will be possible to check

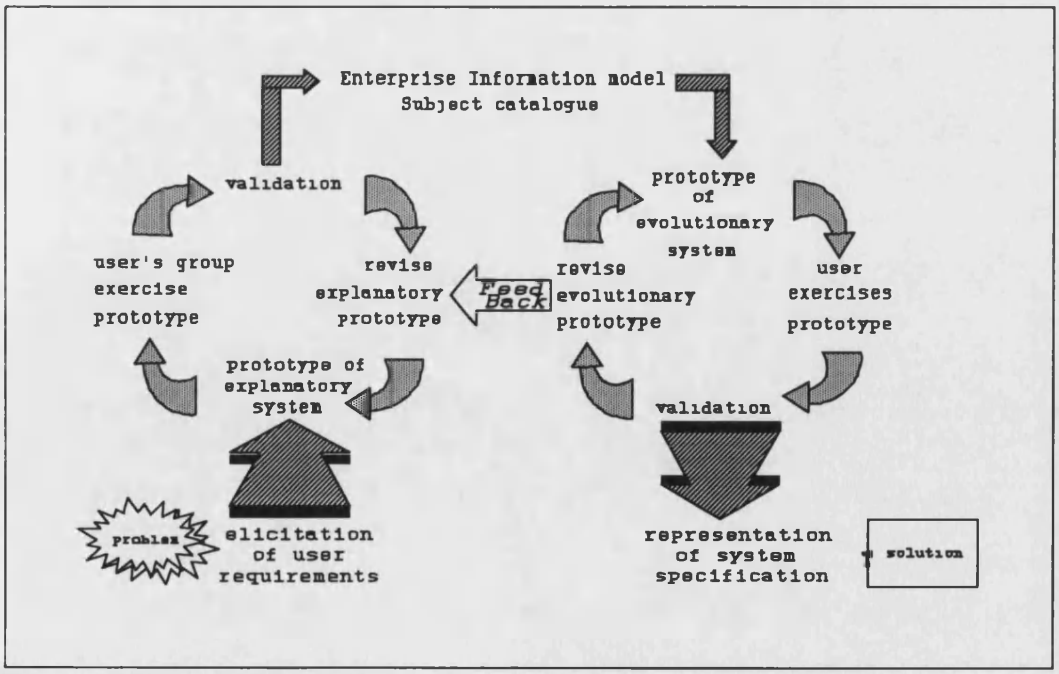


Figure 5.5 The fusion cycle of the proposed development method

the degree of convergence between problem and solution by seeing how closely the explanatory system models the observed behaviour of organisational agents. This feedback process represents the **fusion** cycle. The cyclic processes of requirements determination in the vision cycle and specifications improvement in the action cycle are linked together by the fusion cycle.

We call the first cycle an upstream process or an explanatory process, and the second, a downstream process or an evolutionary process (see figure 5.6). This new perspective on evolutionary development targets a certain development environment that makes it possible to deal explicitly with changes in user requirements. It emphasises that the semantics of user requirements need to be considered with regard to the meanings of actions and behaviours. It aims to provide an explicit focus on a more explanatory manner of working with technical and organisational issues of user requirements, by adopting a collective learning process. Each cycle is a learning process for project members. The explanatory system provides conceptual training for problem understanding and the evolutionary system provides procedural training for solution improvement. Results from the explanatory system are delivered to the evolutionary system. Results are in the form of an enterprise information model which is catalogued by subject clustering. Both explanatory and evolutionary systems have their own cyclical processes to improve their outcomes through active user participation. They also link and converge together via a feedforward and a feedback process. The feedforward process delivers the results from the explanatory system to the evolutionary

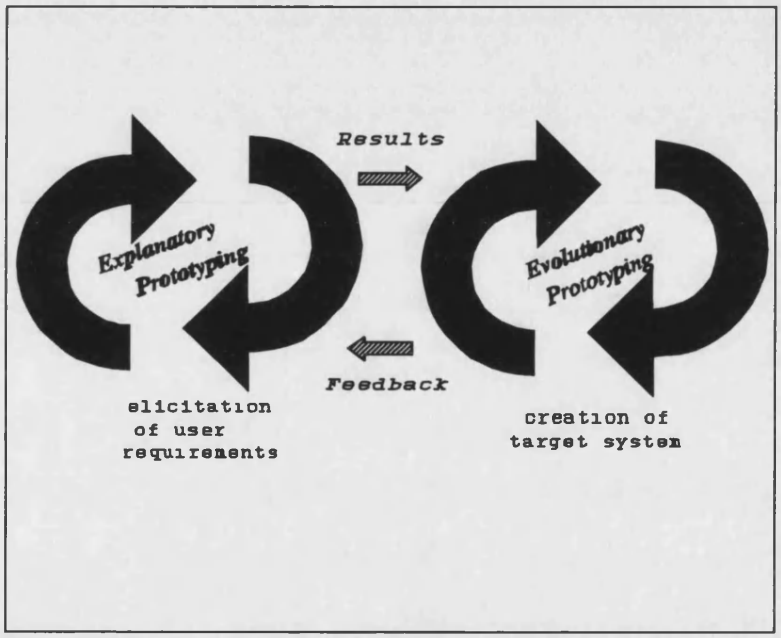


Figure 5.6 Two inter-linked development processes

system. The feedback process reflects the evolutionary system evaluations back to the explanatory system. All processes continuously enrich the user understanding about the system under development. There is no predictable end to these cyclic processes until the adequacy of automated software systems desired by user groups is achieved at a specific time. The convergency of the approach toward the object system is always maintained throughout the development process due to involvement of all responsible agents.

The proposed method examines the wider environment in which the system under study is located. The holistic approach used in the proposed method attempts to establish the entire picture of information requirements, placing them into context within the whole set of organisational norms. It pays more attention to organisational behaviour as a whole than does the technical prototyping approach, which is concerned with specific functions being examined.

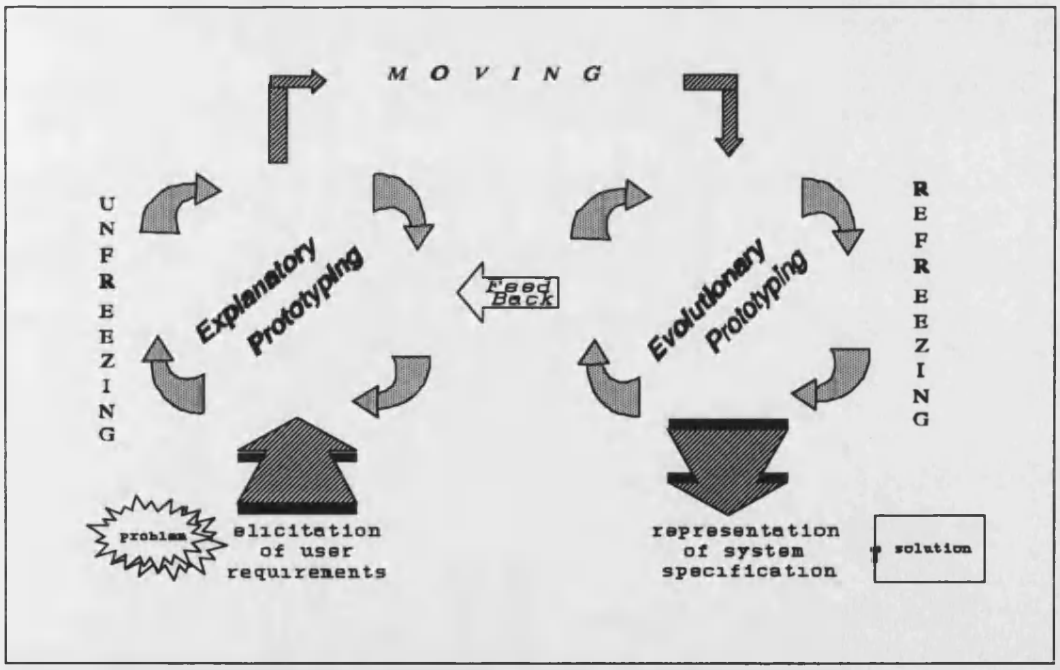


Figure 5.7 Planned change model of the development method

The main idea is to provide a new perspective on problem solving by a tight coupling between an explanatory prototyping environment using a semantic agent-based

modelling on the one side, and an evolutionary development environment using prototyping on the other. At the same time, the development process will be managed by using Lewin's (1952) change process theory, employing the idea of planned organisational change (figure 5.7).

As mentioned above, the vision-action-fusion cycles are the foundation of dynamic development management in the proposed perspective to evolutionary development. It is self-evident that the vision stage, as the point of departure for implementing each cycle, must be carefully drawn up so that it can serve as a secure foundation for the rest of the cycles. It is very important that this cycle is properly established and understood by all participants. Then, if the development starts to deviate from the plan, it is easy to pinpoint the cause of the deviation and to portray each prototype with its original requirements.

5.5.4 A new metaphor for information systems development

We argue that since the information systems development field has both social and technical elements, we need the scientific equivalent of WYSIWIS[1]; we need to have a way of knowing that "what you mean is what I mean". Semantic analysis from the perspective of semiotics provides a way for user and analyst both to find a common designation of the terms they share in their communication. Users may have problems with the interpretation, or they may need to build a shared meaning, in either case they can benefit from a semantic schema as a common referent during the evolution of the new system. We are proposing a new framework for dynamic systems development, connecting information systems interpretation with systems design. While the first cycle is to find a referential base that allows us to develop shared meanings, the second cycle evaluates it through demonstrating the WYSIWYG[2] prototypes (Figure 5.8), the prototypes which offer the actual representation of the target software system. Both loops are integrated toward continuous improvement of the enterprise information model.

---

[1]WYSIWIS: What You See Is What I See

[2]WYSIWYG: What You See Is What You Get

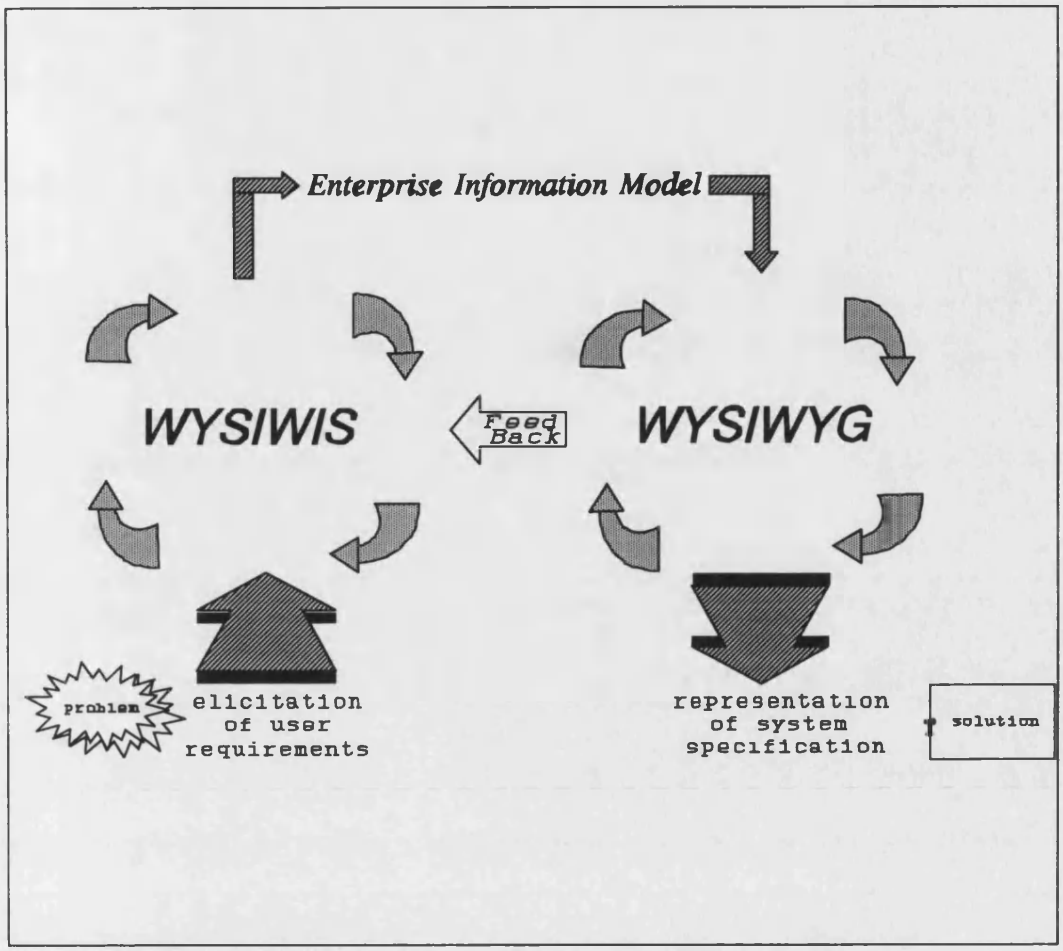


Figure 5.8 Connecting system interpretation to system design

5.6 The proposed development method

Unlike the comprehensive analysis and design methods like SSADM (Structured Systems Analysis and Design Method), the proposed perspective on evolutionary development has no need for a large number of different kinds of charts and specification languages. The ontology charts contain the most stable aspects of the system. The unification provided by this one chart, we expect, remove most of the need for elaborate, bureaucratic documentation which plagues many current methods. The essence of change in the evolutionary approach would not allow bureaucratic documentation even to be possible. So the resulting documentation in the proposed perspective on evolutionary development needs to be minimal in volume but stable and maximally structured as an aid to the understanding of the organisation. The aim of documenting by ontology charts in a form of enterprise information model and subject

catalogue is to provide a positive aid to mutual understanding and communication about any complex collaborative enterprise.

In addition to a new theoretical foundation for evolutionary software prototyping and based on the new perspective on evolutionary development presented in the previous section, a development method is also proposed to formalise the stages and to merge them into a coherent whole. According to the planned change model (Levy, 1986), the following method can be mapped to the three stages of development process. The objective of the following method is not to offer step by step prescriptions in analysis and design. It covers just the most important issues raised in using the proposed perspective. The proposed perspective offers semantic analysis for elicitation of information requirements. The ontology charts provide requirement specifications ready for prototyping. Developers are free to employ their own implementation approach in software prototyping.

Unfreezing stage (Vision cycle):

- ensure that the need for change exists (Bostrom, 1989; Bostrom & Heinen, 1977)
- open the discussion about the goals of the information systems
- define the scope of the system and its relation to the organisation (Ginzberg, 1981)

  These enable one to work systematically from the vaguest problem outlines, possibly from unstructured interviews with those involved in the problem. The analyst can start from a short list of the central goals and relate them to the activities of the business domain. Then through studying the goals and business activities, the vista of the objective information system can be uncovered and also assessed for its impacts. This can be done by discussion with various users of the system who may provide several different but justifiable perspectives. These discussions might be structured. The expected output is the definition of a focal system and specifications of how it begins and ends and what marks its start and finish.

- realign goals of the information system and organisation (Dickson & Simmons, 1970)
- remap the focal system and its collateral systems (Backhouse, 1990)

After preliminary skirmish with the problem, enough common terminology will have been established to enable the central systems of concern to be identified and described. The analyst can then take that central or focal system and relate it to the existing infrastructure. The collateral systems are those that surround the focal system to bring it into existence and give it any value it may have. Each collateral system can be treated as a new focal system, so that the analysis process appears to continue, by an endless recursion. However, the analysis can be terminated when the collateral systems it generates are ones that analyst and user can delegate to the existing infrastructure of the surrounding economic and social systems and to the informal resources of the organisation itself. Focal and collateral systems are the preliminary definitions of subject areas which will take shape in the course of analysis.

- employ semantic analysis for the focal system
  - generate the candidate affordances
  - apply agency structuring
  - exercise ontology charting
- prepare explanatory prototype system of the focal system under consideration

These activities explore in depth what the users mean by all the terminology they have employed in describing their problem domain. They result in a definition of the ontology of a subject area of the problem domain, a very stable structure that accounts for what the users perceive to exist in their world, devoid of information about their organisational behaviour. In order to generate candidate affordances, the analyst needs to take the texts of available interviews or descriptions of the problem as raw materials. He may also add to them relevant documents which the users supply. The process of agency structuring as explained in chapter 4 is an application of the semantic analysis technique to the sub-task of determining which agents (individual and corporate) are involved in the problem domain. It also includes the analysis of the constitution of the corporate agents. The output of this process will be a chart of the agents involved, showing which ones

depend for their existence upon others. Now the analyst can concentrate on ontology charting, the central technique of semantic analysis. The depth of the ontological analysis is potentially infinite but it is limited by the extent to which one wishes to take the description of the organisation.

• decrease the semantic gap (De Brabander & Thiers, 1984) between the use world and the analysis world

- allow user to exercise the explanatory system
- remove other concerns from user
- allow user to express resistance

• define roles and responsibilities

• analyze the norms and proto-norms embedded within the ontology chart

After preparing an explanatory prototype (in a form of an ontology chart of the focal system), it is the time for the user to walk through the prototype and provide feedback about norms governing each part of the ontology chart and exercise the responsibility structure and role definitions expressed in the ontology chart. Upon the stable ontological structure of ontology charts, the analyst will be able to impose information about the rather less stable structure of organisational norms. The least he can do is to state who takes responsibility for deciding the existence of each thing. Next he needs to define what information is relevant to each responsible decision, leaving the agent to decide how to use it. A proto-norm is simply defined as a norm where the condition is only specified as a list of relevant information. The evaluation process is carried out by the person responsible in the light of this relevant information. The proto-norm will consist of a simple list of relevant information for each decision parameter like start, finish or whole existence of an affordance. A normal consequence of this analysis will be to introduce many more candidate affordances and to integrate them into the ontological structure. The formal semantic schema and ontological structure provide inputs to responsibility definition and norm analysis activities. The result is the specification of a constitutional structure and an allocation of individual responsibilities. An explanatory system exercised

by user groups is the best way to find a single frame of reference for requirements specifications. Now the analyst has to plan for an approval of the explanatory prototype system of the focal system.

• provide conceptual training (Ginzberg, 1981) and explain definition of roles and responsibilities to the users (Ginzberg, 1978a; Ginzberg, 1978b)
• seek management commitments (Zmud & Cox, 1979)
• assign user responsibilities (Ginzberg, 1978b)
• seek approval for the ontology chart of the focal system

This step mainly involves validating the ontology chart of the focal system. The validating process of explanatory system encourages all responsible agents to seek for consensus in semantic disagreements before committing to any development of a software prototype system.

• review the subject area clustering catalogue
• assign the validated ontology chart of the focal system to the subject catalogue
• extend the enterprise information model with the validated ontology chart
• seek revision for invalidated parts of the explanatory system
• plan for the software prototype development in the next stage

Movement stage (Action cycle):

• give priority for evolutionary prototyping to the subject areas with most semantic complexity among user groups
• focus on a subject area ready for evolutionary prototyping
• elicit specifications of explanatory system of the part of subject area suitable for automation
• develop an implementation model for the prototype system

At this point, we address the implementation of the software prototype systems. The evolutionary process of software development will allow developers to start development even though not all parts of the enterprise information model are complete. The evolutionary process will provide the users with the opportunity to examine the complex subject areas of the

enterprise information model and provide more feedback from the real working version of the object system. The subject catalogue will maintain the change control and clustering activities in software prototyping and the relationships among different software modules. The development environment and implementation approach are inputs to this stage of development. We believe the semantic agent-based formalism has the capability to furnish its own design and development techniques which need to be investigated in any future development of this research. It is also possible to transform formally the ontology charts to existing design and implementation modelling techniques, similar to object modelling representations. These issues and other implementation concerns will be discussed in chapter 8 of this thesis. The least expectation is that the enterprise information model in the form of ontologically related semantic schemata can be seen as a robust, stable and flexible requirements specification in the most structured manner, which provides viable inputs to any design and implementation approach.

- implement software prototype system
- provide general training to get rid of computer fear
- provide procedural training (Bostrom & Heinen, 1977) to use the evolutionary prototype system
- monitor system and user performance and provide ongoing assistance for users
- plan refreezing the set of system specifications in the next stage

Refreezing stage (Fusion cycle):

- set up feedback systems for users input about the evolutionary system
- integrate the prototype to the evolutionary system
- elicit systems specifications from evolutionary system as verified by users
- revise the implementation model of the evolutionary system
- provide feedbacks to explanatory system from findings of revision in evolutionary system

The unfreezing stage provides valuable feedback to both the evolutionary

and explanatory systems. With the use of the evolutionary system, the user can visualise how much each agent depends upon the action or decisions of others and how much authority is going to transfer to the formal computer system. The analyst will be able to review these dependencies so that the agents, for whom the system is being constructed, can understand better how their actions interrelate. The earlier stages of development will have created an explanation and representation of an actual system which is now treated as an object of investigation. The scale of the business activity specified will be established, the extent to which responsible agents depend upon the decisions of others will be exposed for further discussions and more feedback will be mapped back to the enterprise information model to enrich the whole picture of the substantive business activities. Only the validated parts of the evolutionary system will be amended to the object system which will form the final specification of the envisaged information system and supported by a comprehensive enterprise information model.

- realign goals of information system and organisation (Dickson & Simmons, 1970)
- plan for a new revision to the enterprise information model

The proposed method continuously improves the target system by adopting the necessary changes required at the conceptual and technical levels and maintains growth under the overall pattern of evolution at the organisational level. The above sequence in which the development is carried out is very flexible. The characteristics of the problem in hand and the priorities of the users will determine what work to undertake next. We envisage users and analysts working together to clarify the problem and simultaneously evolve a prototype. The goal is an organic, incremental development of enterprise information model which uses the explanatory prototypes supplemented by re-implementation, on the basis of the evolutionary prototyping approach.

During the unfreezing stage of development, the semantic analysis technique may be regarded as requirements elicitation tool. It is based on the assumption that the business is an infinitely complex social system upon

> which we intend to impose some formality, in order to help it perform more effectively. The scope for analysis, therefore, is never exhausted, but with the users, the analyst must decide when it has gone far enough. Wherever the analytical process stops, the results will be expressed formally in a way that makes explicit where the analysis could have continued. It is as though the specification is a logical structure encrusted with question-marks. Every answer just increases the number of question-marks as well as adding something to the logical structure. The elicitation process never ends but can always pick up the right thread as business requirements enlarge (Stamper, Backhouse, & Althaus, 1989).

5.7 Conclusion

This thesis so far offered an opportunity to examine, in an argumentative/subjective mode, the implementation process of evolving information systems. The most important deficiencies of the evolutionary approach in large scale information systems development projects were identified and a new perspective for this approach was proposed. This perspective attempts to bring together semantic analysis as an analytical technique and prototyping as a technical platform in a logically linked process. The process of requirements determination adopted in the proposed perspective is completely different from the assumptions behind the prototyping approach which mainly focus on determining how the key processes of the organisation contribute to the intended performance outcomes and what data they need for effective functioning. The major objective of analysis in the proposed method is to understand and investigate the existing basis of interaction and communication, such as the differing horizons of meaning of various users. The design process focuses on reconstructing user language to support interaction in order to capture more effectively meanings as conveyed in ordinary user requirements. The proposed perspective aims at increasing mutual understanding and the creation of new meanings and through unfreezing and refreezing stages in evolutionary process facilitates interaction and exchange of information.

The method described in this chapter applies a semantic agent-based formalism to

support incremental development of the enterprise information model. This model is the first representation of the target system in a robust and flexible form, clustered into ontologically related subject areas. Therefore a set of affordances and interconnections between them may constitute each subject area and its corresponding prototype. As a result, the software prototype system can be described using a suitable implementation environment. The description represents a semantic model, and its execution may exhibit operational semantics. The semantic model allows readers of the model to understand its semantics very clearly. During the evolution of the target system, the semantic model is improved repeatedly until it satisfies the user requirements. Then at each cycle of development, the model is improved toward the total enterprise information model. The major characteristics of the application of the semantic-model-based prototyping are the following:

• While requirements are captured in an informal manner, specifications are identified in a more formal manner.

• There is a semantic model detailing a set of behavioral patterns, free from commitment to any particular natural language.

• It is useful for project members to view an understandable semantic model so that they may comprehend the objectives clearly and quickly. The project members include many persons: users, analysts and designers, programmers and management. Before the semantic model of the object system is elaborated, each member may have different understandings, different interpretation of the organisational behaviour. It is important to elicit the semantics held by each individual member before starting to prototype any part of the object system.

• Semantic models are easy to modify. Modifications will be made until all participants are satisfied.

The new method focuses on the importance of language as the principle vehicle through which the construction of reality is mediated. This viewpoint assigns a strong role to

language analysis in conceptual modelling. From the linguistic point of view the object system is not a perception of reality, but the rule system through which communities interact to make sense of and construct reality. These rules pertain to the ways in which language is used (semantics and syntax) and which govern the intentions and effects of language use (pragmatics). Rules which describe the conditions for actions performed in the social system, are permitted to change. This dynamism is a major advantage of the proposed method over the static constraints specified by other data modelling approaches.

The approach to prototyping taken in this research needs to be viewed from a number of angles, with various analyses and other creative thinking processes to ensure that the research result as foreseen is relevant and practical. In the next chapter we will address this issue.

CHAPTER 6

Explanatory empirical study

Chapter Overview

This chapter presents the findings from a case study conducted in one of the "Big Six" management consultancy companies. It examines the relevance of the semantic agent-based modelling formalism for high level corporate modelling. Section 2.1 introduces key findings of other researchers' empirical studies regarding the application of the planned change model to information systems development. These studies provide more insights into the requirements of the planned change model, specifically during the unfreezing and refreezing stages of change process. They emphasize setting frames of reference and bridging the semantic gap between parties involved in design, facilitating conceptual training and institutionalizing the change and finally the importance of role clarification and responsibility negotiation as crucial issues in user involvement in the change process. The explanatory case study demonstrates how the semantic schema supports the above-mentioned requirements and suggests that planned change model and semantic analysis might be in harmony with each other. This harmony supports the justification for the use of the planned change theory and semiotic theory as a new theoretical framework proposed by this thesis for evolutionary prototyping. The case studied is an important part of the corporate data model and explains the substantive business of the management consultancy. Section 6.2 is devoted to a full description of the case study and its findings. It explains the case study in ten subsections. After reviewing the background of the company and corporate data modelling project, agency structuring of the model is introduced in subsection 6.2.1. Each of the nine subsequent subsections shows the application of terms within the company and provides the semantic schema for the eight clusters of the case. The findings of each subsection are presented in the form of observations which show the benefits of semantic modelling for issues like: conceptual training, role clarification and responsibility negotiation, specifically at the unfreezing and refreezing stages of the implementation process. Finally section 6.3 not only concludes this chapter but also finalises the whole research approach designed for this thesis.

6.1 Introduction

This chapter describes the validation stage of the research method. There are a number of empirical studies reporting the application of the planned change model to the information systems development process. During the validation stage of the research approach, we reviewed these secondary studies to explore the practical adequacy of the planned change model. The aims are to relate the findings of the explanatory case study to the requirements of the planned change model and to show the harmony that exists between support theories in the proposed method.

The Lewin/Schein process theory of change (Lewin, 1952; Schien, 1961, 1972) has been employed in number of empirical studies as the basic model for research on the implementation process. This planned change model, particularly as elaborated by Kolb and Frohman (1970), suggests key issues which must be resolved if the change effort is to succeed. Among these issues are institutionalizing the change during refreezing stage of development (Zand & Sorenson, 1975; Goodstein & Bruke, 1991), setting frames of reference and bridging semantic gaps through better communication between users (De Brabander & Thiers, 1984), providing a medium for conceptual training at an early stage of implementation process (Ginzberg, 1981) and considering role clarification and responsibility negotiation of different agents involved in the change process (Bostrom & Heinen, 1977). The results of different researches indicate that the resolution of these issues is important to the success of information systems implementation efforts.

The explanatory case study explains the power, practicality and ability of semantic analysis to respond to the above-mentioned requirements of the planned change model in coordinating the unfreezing and refreezing stages of the change process. The case study explains how the semantic agent-based formalism supports the representation of social and business affairs in the form of a corporate data modelling case. The case study supports subjectively the justification of the important analysis technique suggested in the proposed method. It also addresses the issues explored by exploratory case study, discussed earlier in chapter 4, regarding the shortcomings of evolutionary prototyping approach in information systems development.

6.2 Explanatory case study: corporate data model

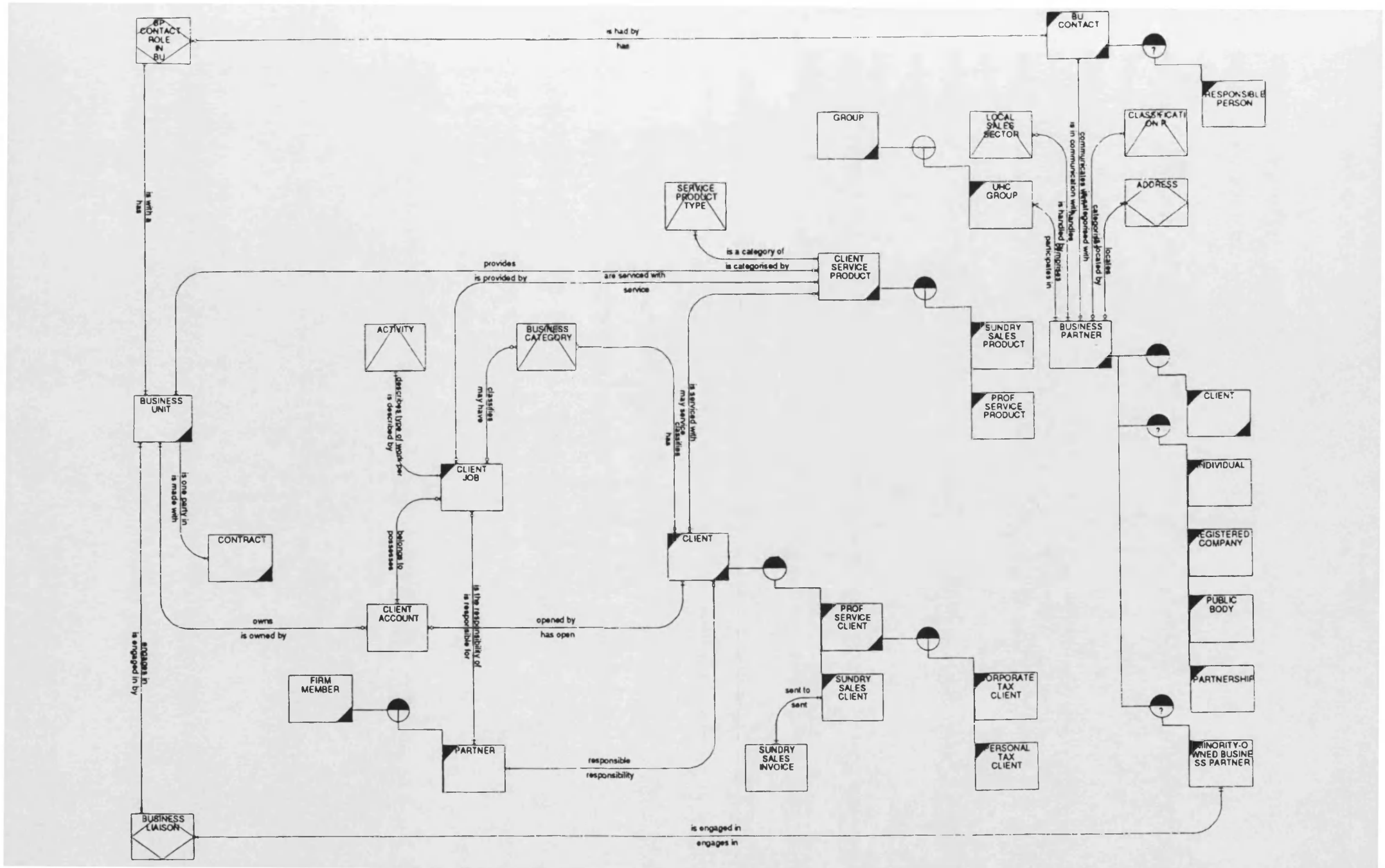
The proposed method by this thesis suggests a planned change model for managing the implementation process. The previous section mentioned the important issues in the application of the model to the evolutionary information systems development. The analysis of those issues provides insight into the characteristics required from a semantic tool during the unfreezing and refreezing stages of development. The proposed method in this thesis suggests semantic analysis techniques to be employed during those stages. The features of this technique discussed earlier in chapter 3 appear to respond well to the issues mentioned above in previous section. We have also conducted a case study to provide further justification for the application of semantic analysis and for its characteristics in considering the shortcomings of evolutionary prototyping and issues related to the planned change model. This explanatory case study is the subject of present section.

The subject of the explanatory case study is the corporate data model of one of the biggest management consultancy firms. The firm is run as a partnership with a headquarters based in United States, and the UK partnership is part of the international partnership of the firm. The UK partnership has a corporate data division which provides consultancy and support for the development of the corporate data model for large enterprise. In order to have a corporate data model reconciled across the whole partnership all around the world, the corporate data division of the UK partnership in 1991 started a project to develop such a data model. The project, which is coordinated by the US headquarters, has the objective of completing a corporate data model for each partnership in each country and then of merging them all. The focus of this case study is the corporate data model project of the UK partnership. The project is one of the first projects of its own and was started five years ago with the direct supervision of the head of corporate data division as project manager, together with a senior analyst from that division. The project at the time of study in November 1995 was still ongoing, despite the fact that both above-mentioned persons responsible for the project are now handling the project on a part-time basis, owing to the pressure of other consultancy projects with their clients.

The corporate data model of the UK partnership was developed using entity-relationship (ER) modelling. It now consists of tens of different entities with very complex relationships between them. Understanding the model as a whole is extremely difficult and without the assistance of the project manager an impossible task. One of the most important entities within the model is client since the whole organisational resources and activities of the firm are directed towards its clients. The focus of the case study has been restricted to this entity and all other entities with direct connections to it. This restriction resulted in a (sub)model of around forty entities, shown on the next page.

The model is also supported by a comprehensive data dictionary which defines different entities and clarifies their relationships to each other. The case study employed existing data analysis documents and in-depth discussions with the senior analyst. The result of the study was a very comprehensive report, containing the semantic schema developed on the existing model and the definitions provided in data dictionary. The report was then presented to the project manager, senior analyst and another person from the rules and regulation division of the firm. The reaction of different persons to each part of the report and discussions stimulated by the report are also considered as part of the study of the case. The explanation of the model is discussed in the following ten subsections and the findings from each subsection are presented in the form of observations. The observations support the justification of the semantic agent-based modelling and its power, effectiveness and practicality in relation to its purpose as an explanatory prototyping tool in the proposed method of this research.

This section consists of ten subsections. Subsection 6.2.1 introduces the responsible agents and the other nine subsections correspond to the different clusters of the semantic schema developed for the case. In each subsection, the original corporate data model, definitions of related entities and ontology chart of each specific cluster of the semantic schema will be presented. Dividing the model into eight clusters is the result of the natural decomposition characteristic of the semantic agent-based modelling and not of a clustering superimposed for the sake of reducing complexity. Those entities which are considered semantically appropriate for the ontology chart of each cluster are highlighted by shadow colour in each subsection.


BP CONTACT ROLE IN BU
is had by
has
BU CONTACT
RESPONSIBLE PERSON
GROUP
LOCAL SALES SECTOR
CLASSIFICATION
UHC GROUP
ADDRESS
SERVICE PRODUCT TYPE
is a category of
is categorised by
CLIENT SERVICE PRODUCT
provides
is provided by
are serviced with
service
is with a
has
communicates with
is categorised with
is handled by
participates in
locates
is located by
ACTIVITY
BUSINESS CATEGORY
SUNDRY SALES PRODUCT
BUSINESS PARTNER
PROF SERVICE PRODUCT
describes the kind of work done
is carried out by
classifies
may have
is serviced with
may service
has
BUSINESS UNIT
CLIENT
INDIVIDUAL
CLIENT JOB
is one party in
is made with
CONTRACT
belongs to
possesses
CLIENT
REGISTERED COMPANY
PUBLIC BODY
owns
is owned by
CLIENT ACCOUNT
is the responsibility of
is responsible for
opened by
has open
PROF SERVICE CLIENT
PARTNERSHIP
FIRM MEMBER
sent to
sent
SUNDRY SALES CLIENT
CORPORATE TAX CLIENT
PARTNER
responsible
responsibility
SUNDRY SALES INVOICE
PERSONAL TAX CLIENT
MINORITY-OWNED BUSINESS PARTNER
is engaged by
engages
BUSINESS LIAISON
is engaged in
engages in

6.2.1 Agency structuring

Agents are special types of unit of analysis, as they hold responsibility in the world of actions. Therefore identification of agents and their names in normal usage in the organisation is the first task in semantic analysis. It is interesting to know that in the case studied, we only have two responsible agents. One , obviously, the firm itself as a partnership and the other a legal person which by definition might be an individual person or differing types of business units. Any further analysis will follow with the involvement of one or both agents. A legal person might take number of roles in its relationship with partnership, as its client with a contract in between or as a staff member in an employment relationship. To avoiding using the role names, figure 6.1 shows the root names of two responsible agents of the case. It is interesting to see that there are different entities in the original ER model representing different roles of legal person, but there is no entity for the representation of the partnership itself.

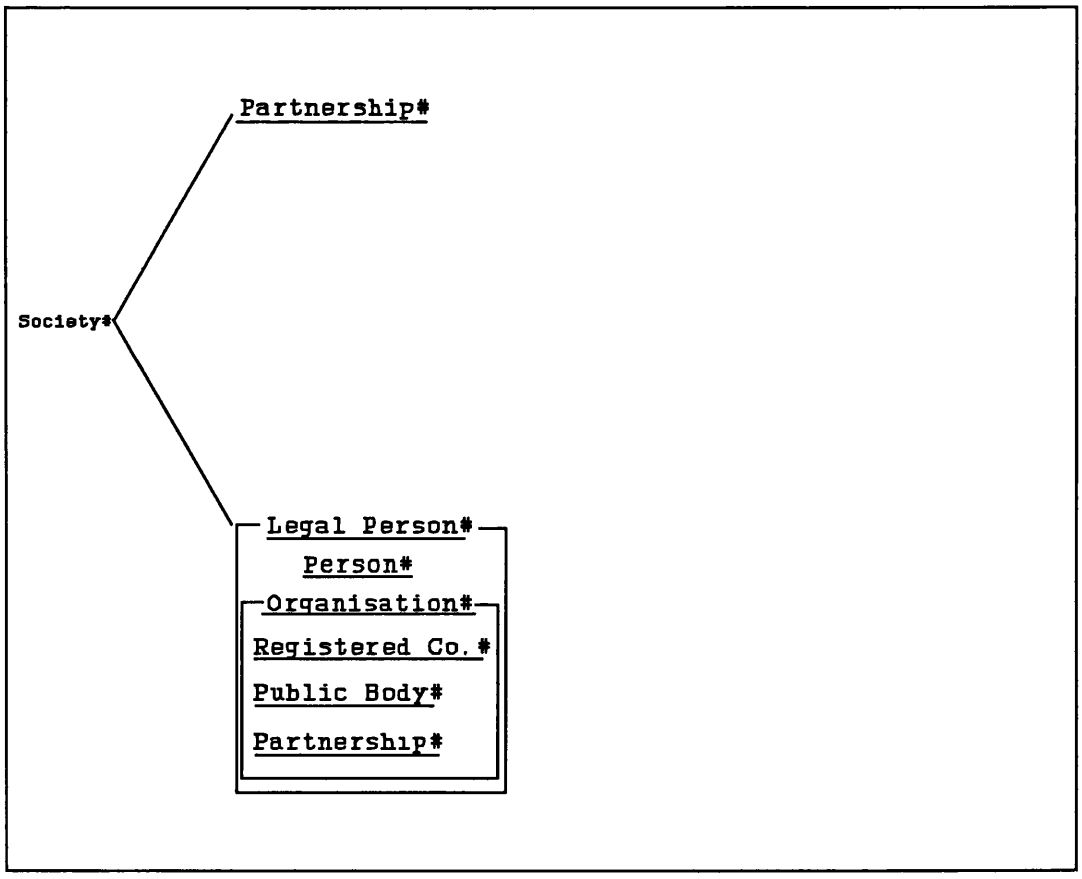


Figure 6.1 Agency structuring

Although a vast amount of responsibility lies with the partnership, the analysis of the corporate model regards it as container of all entities and not an integral part of the model.

> ***Observation 1:*** *Semantic analysis technique has the advantage of identifying responsible agents for placing responsibility. By enforcing a subjective view of the world, it seems impossible to develop ontology charts without identifying the knowing agents who construct the reality through their actions.*

6.2.2 Firm Member

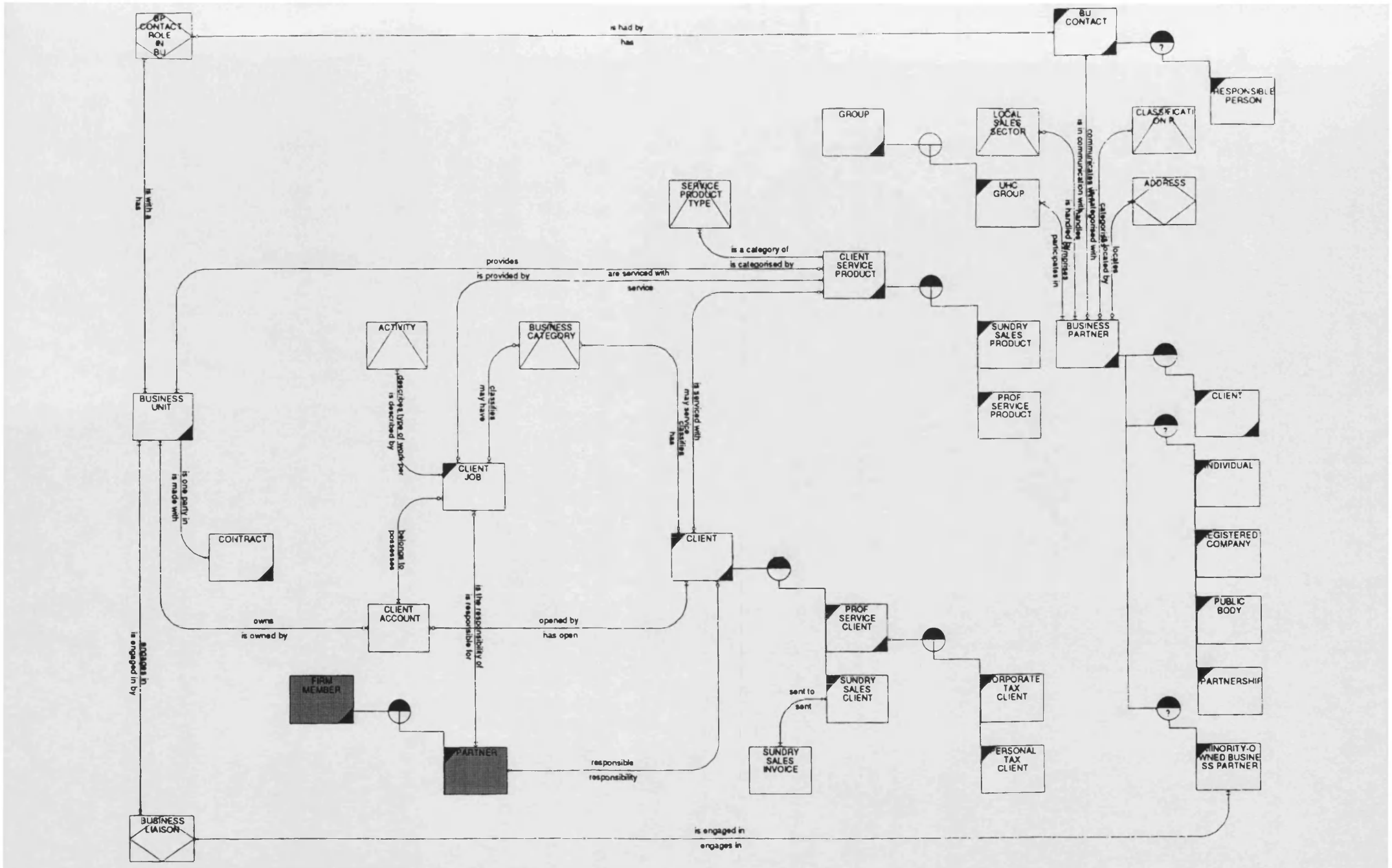
This subsection provides the first cluster ontology chart of the semantic schema, concerning the definition of firm member and its different types. The following are definitions for these names which are simply different roles for a person who has a type of contractual relationship with the firm. The figure in the next page highlights the entities involved in this cluster.

Firm Member: An individual who carries out, or has carried out, day to day activities in support of the business. This category includes **Partners, Staff Members, Contractors, Temps & Permanent Temps.** It excludes those where the contract buys the service rather than a person. eg. ancillary and temporary services such as window cleaning, vending machines attendants.

Partner: A Firm Member who has been elected to the UK partnership. A **Partner** is jointly liable for the debts of the firm & can sign legal documents on behalf of the partnership. Classically a **Partner** is a participant in partnership profit & loss.

Partnership: A group of people who are party to a partnership agreement.

Figure 6.2 shows how the role name firm member arises when there is a form of contractual relationship between a person and the partnership. Contractual relationship as a generic name may have different specifics depending on the type of contract between the firm and a person. For example as it is shown in figure 6.3, if the contract is employment then firm member could be more specifically relabelled as staff member, if the contract between the firm and a person is a form of partnership contract then


BP CONTACT ROLE IN BU
is had by
has
BU CONTACT
RESPONSIBLE PERSON
GROUP
LOCAL SALES SECTOR
CLASSIFICATION
UHC GROUP
ADDRESS
SERVICE PRODUCT TYPE
is a category of
is categorised by
provides
is provided by
are serviced with
service
CLIENT SERVICE PRODUCT
SUNDRY SALES PRODUCT
PROF SERVICE PRODUCT
BUSINESS PARTNER
ACTIVITY
BUSINESS CATEGORY
BUSINESS UNIT
CLIENT JOB
CONTRACT
CLIENT
CLIENT ACCOUNT
PROF SERVICE CLIENT
SUNDRY SALES CLIENT
CORPORATE TAX CLIENT
PERSONAL TAX CLIENT
SUNDRY SALES INVOICE
FIRM MEMBER
PARTNER
BUSINESS LIAISON
INDIVIDUAL
REGISTERED COMPANY
PUBLIC BODY
PARTNERSHIP
MINORITY-OWNED BUSINESS PARTNER
is engaged in
engages in
owns
is owned by
opened by
has open
responsible
responsibility
sent to
sent

the role name for the person will be partner and if it is just a general contract the role name will be relabelled simply as contractor, etc.

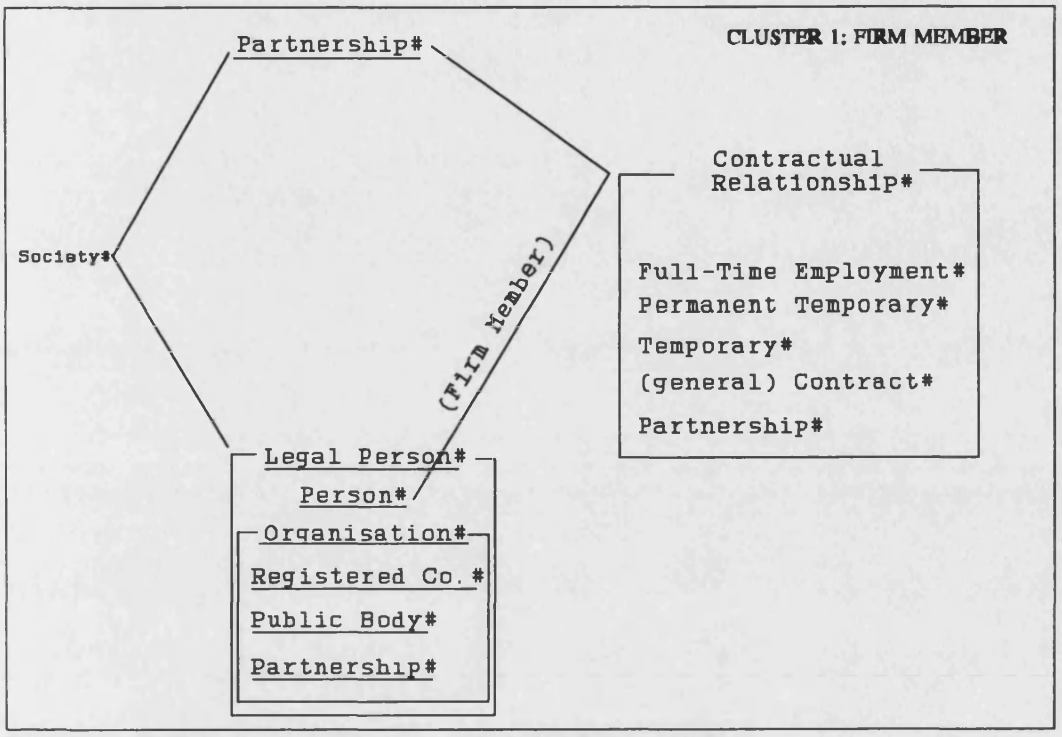


Figure 6.2 Cluster 1: Ontology chart for firm member

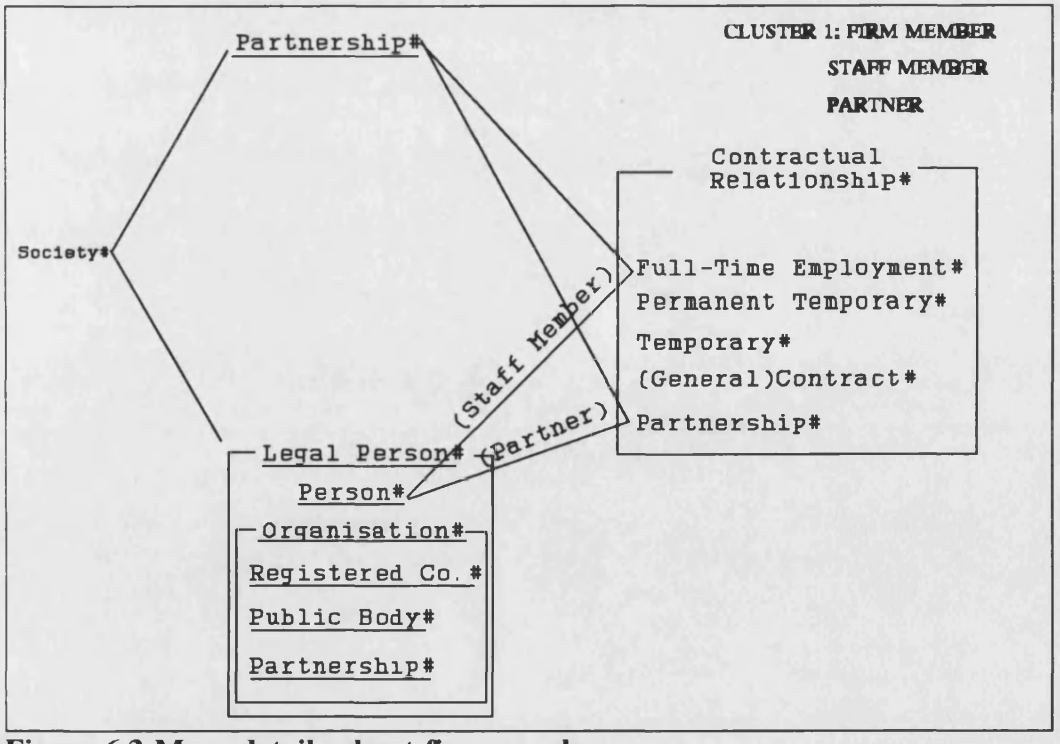


Figure 6.3 More details about firm member

Observation 2: *The generic-specific relationship and its handling of changing role names brings economy of expression to the ontology charting. It also provides consistency as the range of activities expand. For a new type of contractual relationship, we just need to add a new specific to the chart.*

Observation 3: *One problem with entity modelling is how to decide where to begin reading the chart, or before that, where to begin the representation. Reading from left to right, the ontology charts produced by semantic agent-based formalism indicate the pattern of behaviour and actions that needs to be realised before anything on the right hand side can be brought into existence.*

Observation 4: *Ontological dependency in ontology chart can provide a reliable and durable orientation on which to anchor the concepts in the semantic representation. If a person dies, the termination of his contractual relationship is automatically enforced by the semantics of the ontology chart and the termination procedure can be automatically triggered by system.*

Observation 5: *There is also a time constraint implied by ontological relationship. Any concept which has a current realisation can only exist if its antecedents continue to exist. The responsibility for determining whether the realisation has ceased, for example when an employment contract has been terminated, belongs to a responsible agent within the firm who is the guardian of that particular signification.*

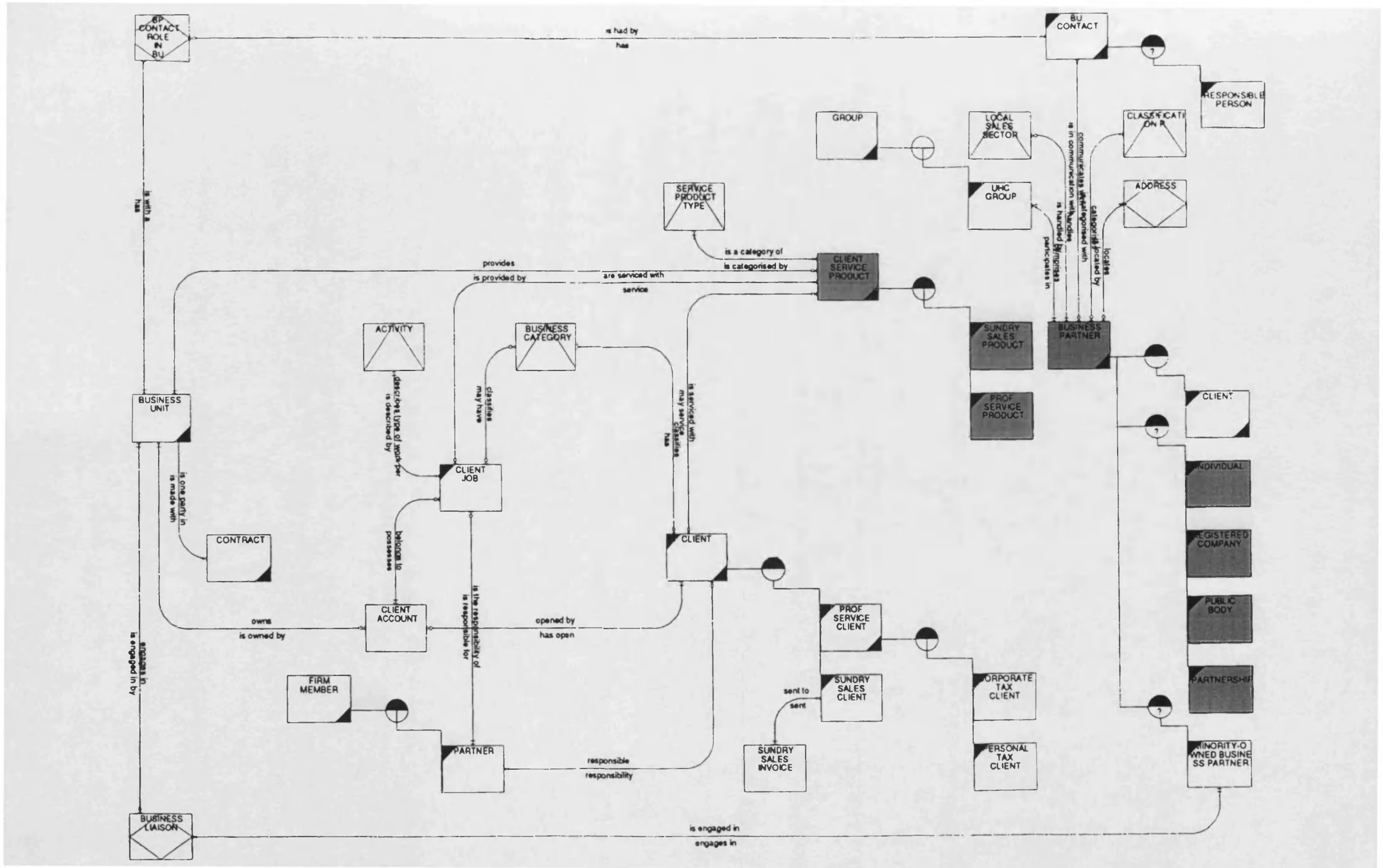
Observation 2 addresses one of the important issues studied by Bostrom and Heinen (1977) regarding responsibility and role clarification from a change theory perspective. They argued that designers can only facilitate individual change through the handling of the change process, as only the individual himself is capable of changing his own behaviour. A semantic schema gives the users and developers the opportunity to discuss and clarify future roles and responsibilities. This aspect of the representation of roles in semantic schema was one of the most attractive ones for the senior analyst during our discussions of the case. The distinction between agents and their roles in different relationships is clearly useful when compared with the treatment of ER modelling for roles as separate entities.

Observation 3 promotes the readability of the semantic schema. The project manager and the senior analyst found the representation of the semantic schema bulky and not economical. But they all agreed that this can be resolved by a computer aided system in support of the modelling technique. The reward is readability of the ontology charts by different users, for whom the charts appeared to be supportive and self-explanatory.

The issue of understandability of requirements representation was also discussed in the exploratory case study reported in chapter 4. The lack of an understandable communication medium between business analysts and User Forum in general and inability of ER modelling representation to furnish such a platform for communication, in particular, were the major findings of that case study. Through clear representation of the users' language in the workplace, we will be able to facilitate design understanding. The concept of time and the question of who decides the temporal constraints of a current realisation were very difficult to convey during our discussions. This is mainly because of our preoccupation with the concept that time can be always kept out of the database structure and has effects just through transactions of the system.

6.2.3 Business partner

The following figure highlights the entities involved in the definition of business partner.


BP CONTACT ROLE IN BU
is had by
has
BU CONTACT
RESPONSIBLE PERSON
GROUP
LOCAL SALES SECTOR
CLASSIFICATION
UMC GROUP
ADDRESS
SERVICE PRODUCT TYPE
is a category of
is categorised by
provides
is provided by
are serviced with
service
CLIENT SERVICE PRODUCT
SUNDRY SALES PRODUCT
PROF SERVICE PRODUCT
BUSINESS PARTNER
ACTIVITY
BUSINESS CATEGORY
BUSINESS UNIT
CLIENT
INDIVIDUAL
REGISTERED COMPANY
PUBLIC BODY
PARTNERSHIP
MINORITY-OWNED BUSINESS PARTNER
CLIENT JOB
CONTRACT
CLIENT
CLIENT ACCOUNT
owns
is owned by
opened by
has open
PROF SERVICE CLIENT
SUNDRY SALES CLIENT
CORPORATE TAX CLIENT
PERSONAL TAX CLIENT
FIRM MEMBER
PARTNER
responsible
responsibility
sent to
sent
SUNDRY SALES INVOICE
BUSINESS LIAISON
is engaged in
engages in

The definition of entities involved in this cluster based on data dictionary are as follows:

Individual: A person

Public Body: A non-privately owned or commercial enterprise.

Registered Company: A company that is registered with Companies House.

Business Partner: A legally explicit & named-part of an organisation, or individual, which has engaged or is a prospect for the engagement of the firm's services. It is that part of the organisation which requires to be identified for firm's operational management.

Client Service Product: A service or product which the Firm proposes or contracts to provide a **Client**.

Professional Service Product: A service which the Firm provides to a **Client** which typically consists of hours plus expenses.

Sundry Sales Product: An item or service offered by the Firm but which is not classed as a **Professional Service Product**. A **sundry Sales Product** is never subject to professional clearances or other **Regulatory Body** imposed restrictions. Example include publications, software and public training courses and conferences.

The distinction between business partner and client as two entities was confused. In the corporate data model, as shown in the previous page, client is treated as a sub-type of business partner, itself a super-type. This problem always arises when we consider role names as real entities and not as based on their root names and the conditions that represent each specific role. In fact business partner is a legal person who gives definition for every service product the firm is providing. The range of service products provided by the firm as a management consultancy can only be realised when a legal person is recognised as a business partner of the firm's partnership. This form of ontological dependency is presented in figure 6.4 marking the second cluster of semantic schema. In the semantic agent-based formalism the underlying notion is that the world is not given, but instead is created by the responsible agents who shape and govern it.

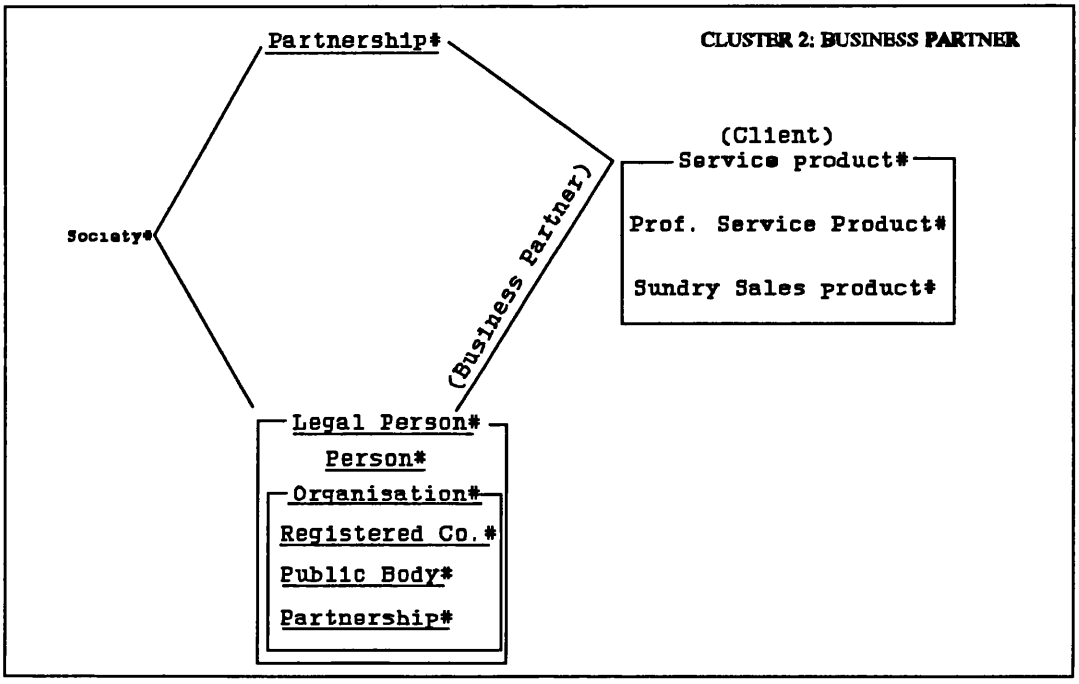


Figure 6.4 Cluster 2: Ontology chart for business partner

The pattern of behaviour that analysts seek to capture in an enterprise model embraces a complete repertoire of organisational activity - what it needs to do to maintain itself and to prosper. These modelling objects, physical and abstract, are ultimately defined not by some form of words, although this helps, but by the activities performed to instantiate them. The precise meaning of business partner, service products or client will be fashioned by how company behaves towards such creations, and since an organisation is composed of many responsible sub-parts, many competing significances will be operating simultaneously and in different periods of time. The formalism used for capturing these requirements needs to be able to reflect this state of affairs. The clarification between the meaning of business partner and client was confirmed through our discussions with project manager and the person from rules and regulation division.

> ***Observation 6:*** *A formalism for representing the information used by the business needs to be able to account for different understandings. Naming and meaning are separate although related ideas. Semantic analysis requires us to record what meanings were in use and at what period of time.*

6.2.4 Client

The next page model presents the highlighted entities involved in the definition of client. The following are complete definitions for those entities from data dictionary:

Business Unit (BU): A time-defined component of the firm which controls resources (including assets). This currently is viewed as a management unit for which the firm determines the Business Objective, and for whom the **Partners** are responsible for the financial reporting and to the relevant Regulator for the conduct of their professional service. **A Business Unit** has a stated **Business Objective** and may have a **Market Specialisation** eg. MCS has Manufacturing Systems & Supply Chain. **A Business Unit** may control resource/assets; be a recognised aggregation point of resource/asset (reporting level).

Client: An **individual** or the legally explicit & named part of an **Organisation** (or that part which requires to be identified for firm's operational management) with which the firm is contracted to provide services/products.

Contract: A legal agreement between two parties.

The following ontology chart marks the third cluster of semantic schema for client.

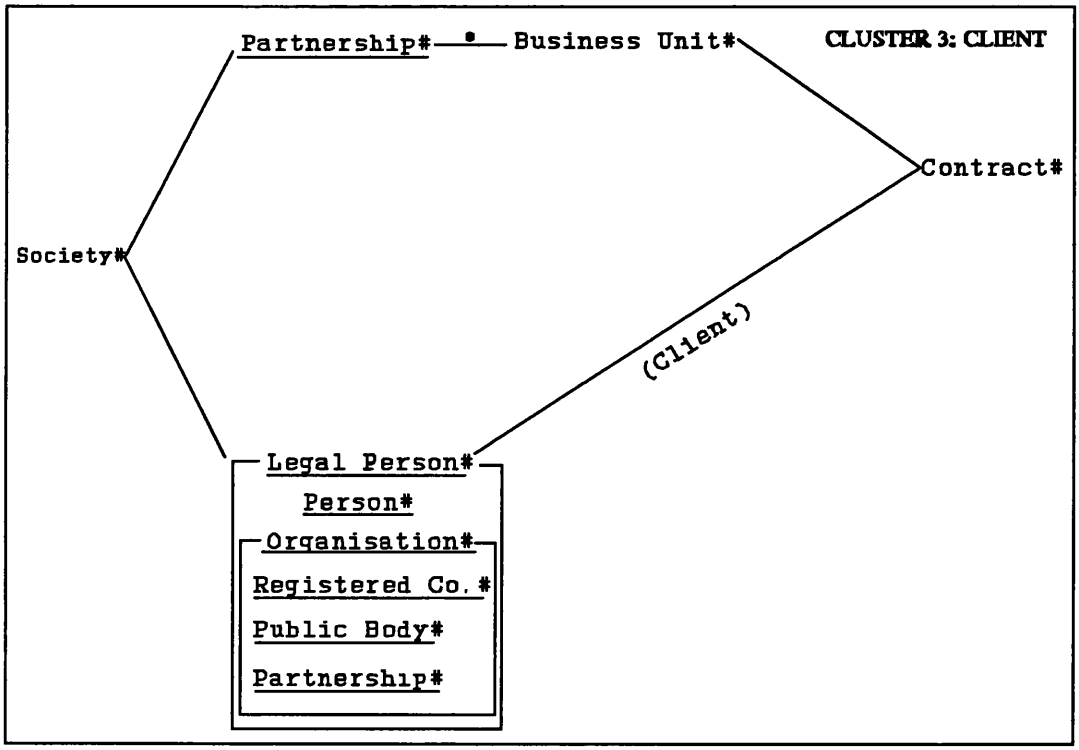
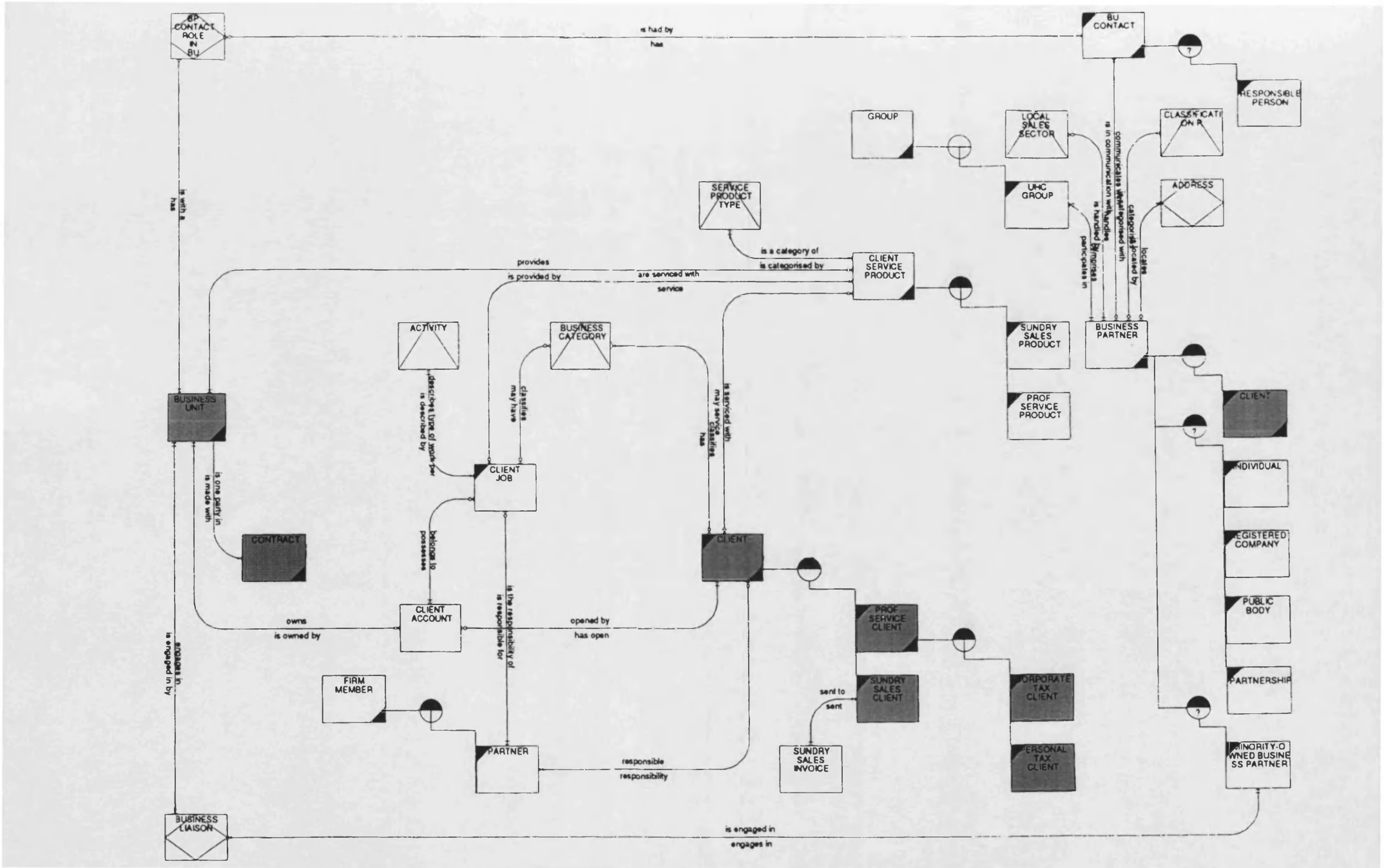


Figure 6.5 Cluster 3: Ontology chart for client


BP CONTACT ROLE IN BU
is had by
has
BU CONTACT
RESPONSIBLE PERSON
GROUP
LOCAL SALES SECTOR
CLASSIFICATION R
UHC GROUP
ADDRESS
SERVICE PRODUCT TYPE
is a category of
is categorised by
provides
is provided by
are serviced with
service
CLIENT SERVICE PRODUCT
SUNDRY SALES PRODUCT
PROF SERVICE PRODUCT
BUSINESS PARTNER
ACTIVITY
BUSINESS CATEGORY
BUSINESS UNIT
describes type of work done
is described by
classifies
may have
is serviced with
may service classes
has
CLIENT JOB
CLIENT
INDIVIDUAL
is one party in
is made with
CONTRACT
belongs to
possesses
is the responsible for
is responsible for
owns
is owned by
CLIENT ACCOUNT
opened by
has open
PROF SERVICE CLIENT
REGISTERED COMPANY
PUBLIC BODY
FIRM MEMBER
SUNDRY SALES CLIENT
CORPORATE TAX CLIENT
PARTNERSHIP
sent to
sent
PARTNER
responsible
responsibility
SUNDRY SALES INVOICE
PERSONAL TAX CLIENT
MINORITY-OWNED BUSINESS PARTNER
is engaged in by
BUSINESS LIAISON
is engaged in
engages in

According to definitions from data dictionary, we can understand that there are different types of client:

Prof Service Client: A **Client** who receives a **Professional Service Product.**

Personal Tax Client: An **Individual** who is a **tax Client** of the Firm.

Corporate Tax Client: A **client** of the Corporate Tax area.

Sundry Sales Client: A customer of one or more firm's **Sundry Product,** who at that time was not a **Client** of **Professional Service products.**

By taking into account different types of client, figure 6.6 presents a more detailed ontology chart of client.

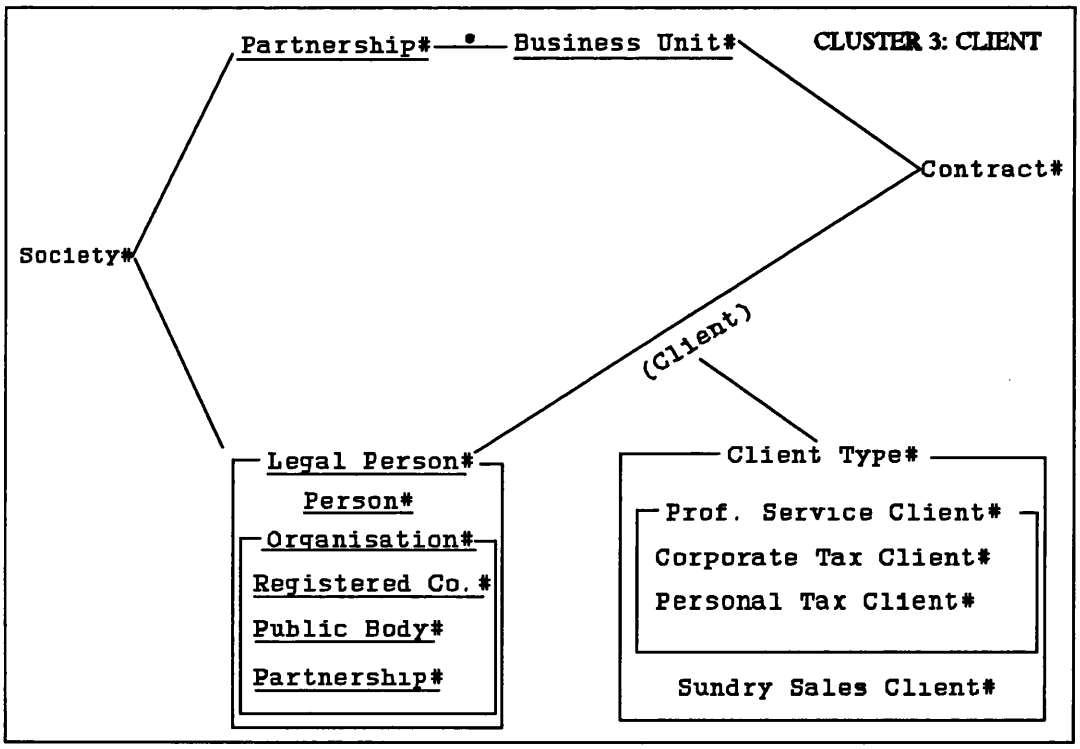


Figure 6.6 More details about client and client types

Itis interesting to consider that while client has no meaning without having a contract with business unit as part of the partnership, and while it is clearly represented in the previous page ER model that business unit **is one party of** the contract, there is no reference to the second party in the contract between client and business unit. It is

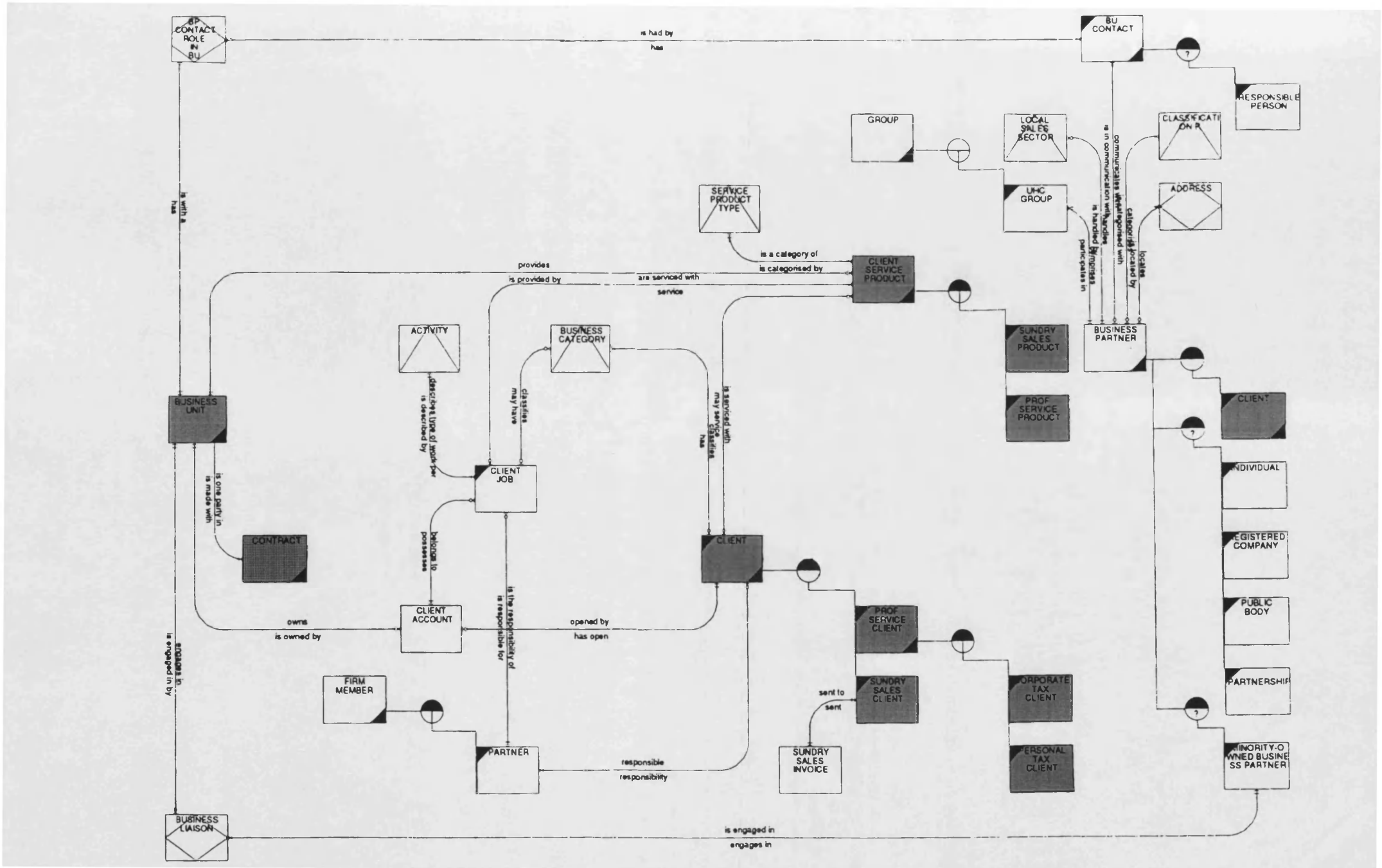
mainly because there are no semantic constraints enforced in ER modelling. This is a typical problem of all conventional modelling approaches in that they assume the world to be made up of ready made entities or objects, which merely need to be articulated.

> ***Observation 7:*** *In semantic agent-based formalism, the precise meaning of terms like client is fashioned by how the firm behaves towards such a creation. In this way it seems very difficult that semantic constraint employed in the formalism neglects the participants responsible for that creation of meaning.*

Further attention to the next page model and the relationship between client and client service products shows how two clusters need to relate to each other in order to create the concept of how client is serviced with (client) service products. Figure 6.7 shows that relationship.

To be more precise, figure 6.8 shows how different types of client are serviced by different types of (client) service products. This relationship is exactly where the relationship between business partner and client becomes clear. Business partner is responsible for creating the meaning for any form of service product. As soon as business partner starts to be serviced by the service product under the terms and conditions of a contract, then he will be also regarded as a client of the firm who is receiving a (client) service product. Only through semantic scrutiny administered in a semantic formalism, we can be sure that namings and meanings precisely refer to each other.

When the contract between firm and client is terminated, the client still can be regarded as one of the business partners of the firm who was, and still can be, responsible in the creation of a specific service product. Figure 6.8 and the review of the ontology charts for the client from figure 6.5 through 6.8 provide more findings from the case study in the form of the next observations:


BP CONTACT ROLE IN BU
is had by
has
BU CONTACT
RESPONSIBLE PERSON
GROUP
LOCAL SALES SECTOR
CLASSIFICATION R
UHC GROUP
ADDRESS
SERVICE PRODUCT TYPE
is a category of
is categorised by
provides
is provided by
are serviced with
service
CLIENT SERVICE PRODUCT
SUNDRY SALES PRODUCT
PROF SERVICE PRODUCT
BUSINESS PARTNER
ACTIVITY
BUSINESS CATEGORY
BUSINESS UNIT
CLIENT JOB
CONTRACT
CLIENT
CLIENT ACCOUNT
owns
is owned by
opened by
has open
PROF SERVICE CLIENT
SUNDRY SALES CLIENT
CORPORATE TAX CLIENT
PERSONAL TAX CLIENT
FIRM MEMBER
PARTNER
responsible
responsibility
SUNDRY SALES INVOICE
CLIENT
INDIVIDUAL
REGISTERED COMPANY
PUBLIC BODY
PARTNERSHIP
MINORITY-OWNED BUSINESS PARTNER
BUSINESS LIAISON
is engaged in
engages in

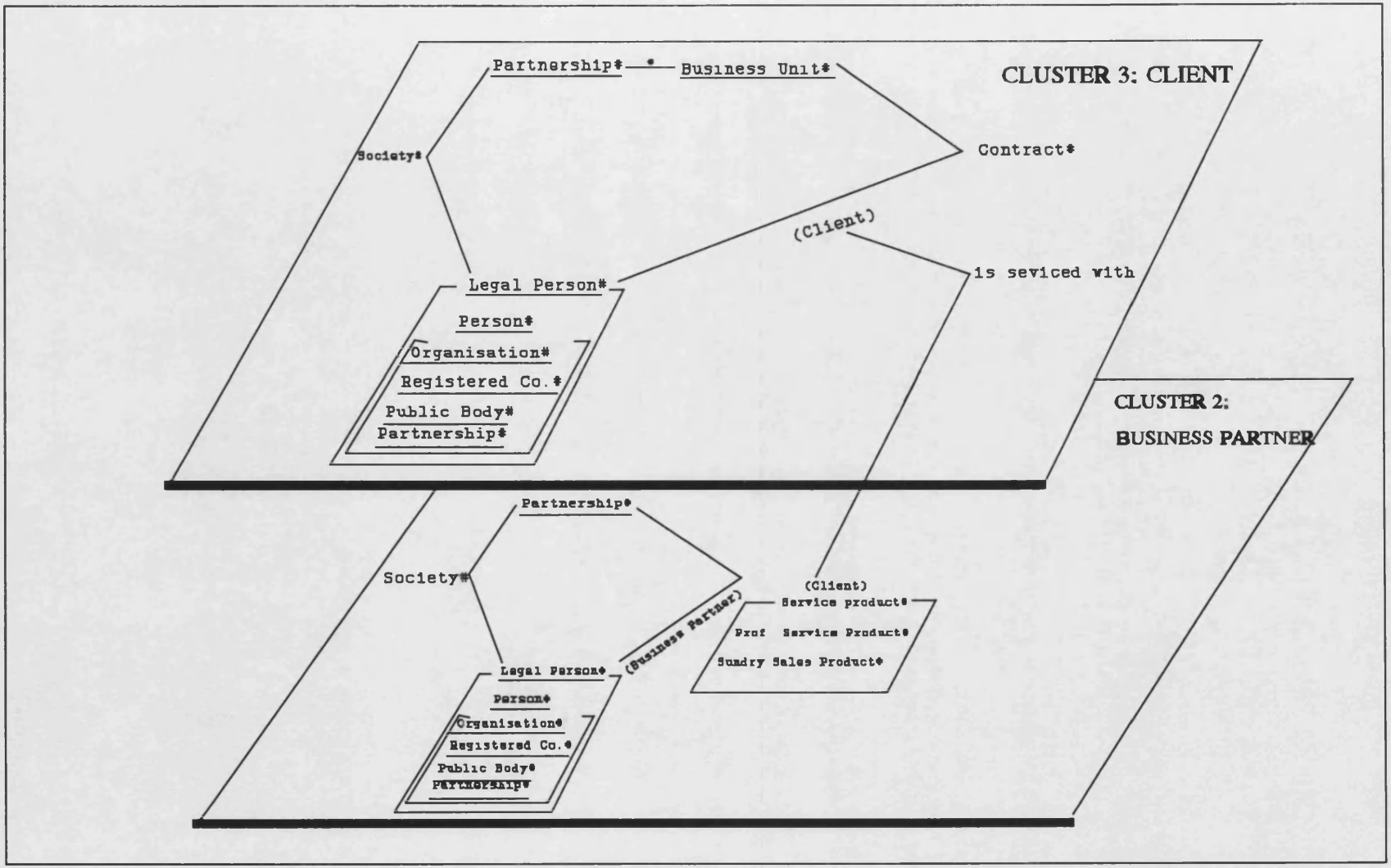


Figure 6.7 Connection between client and (client) service products

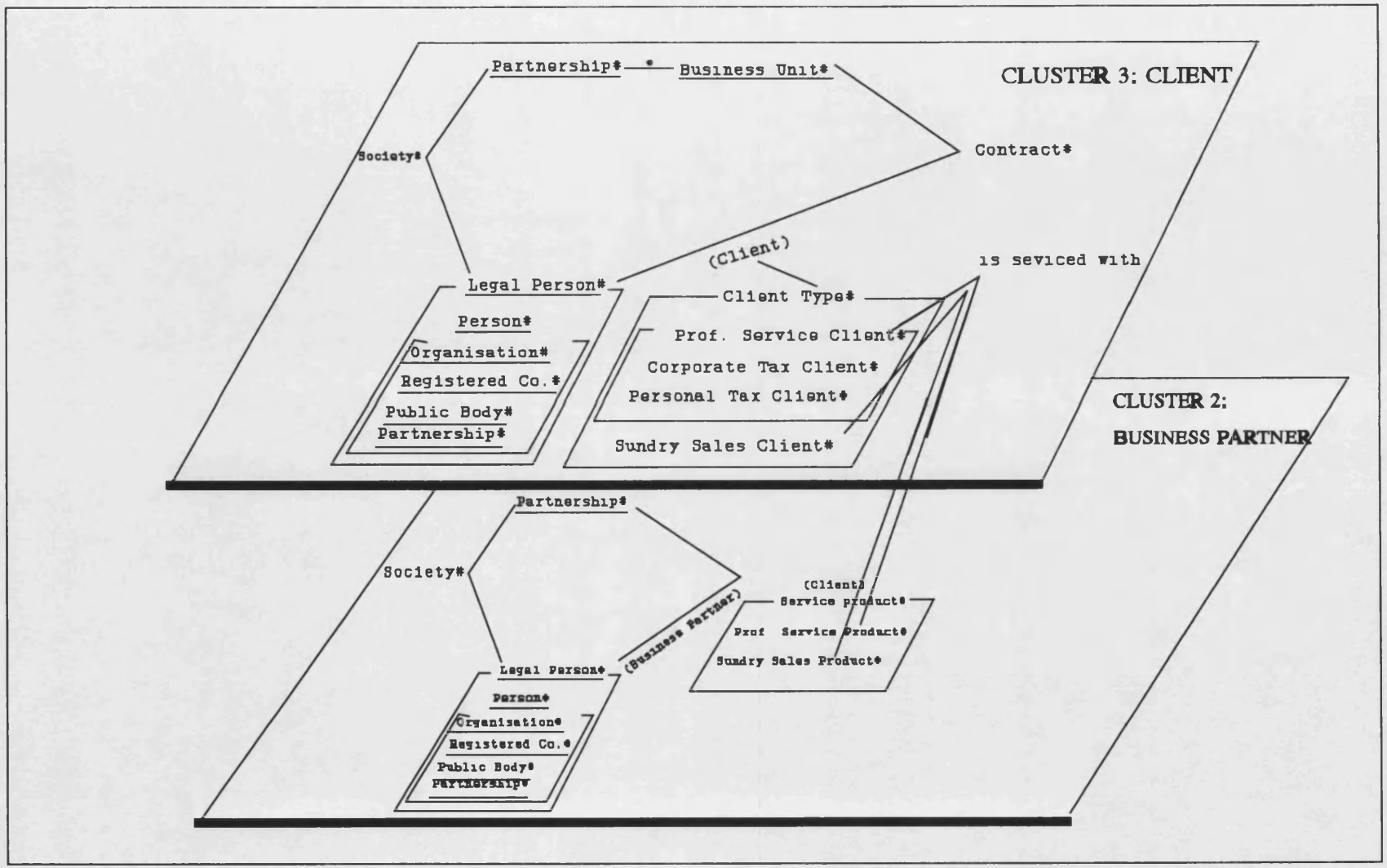


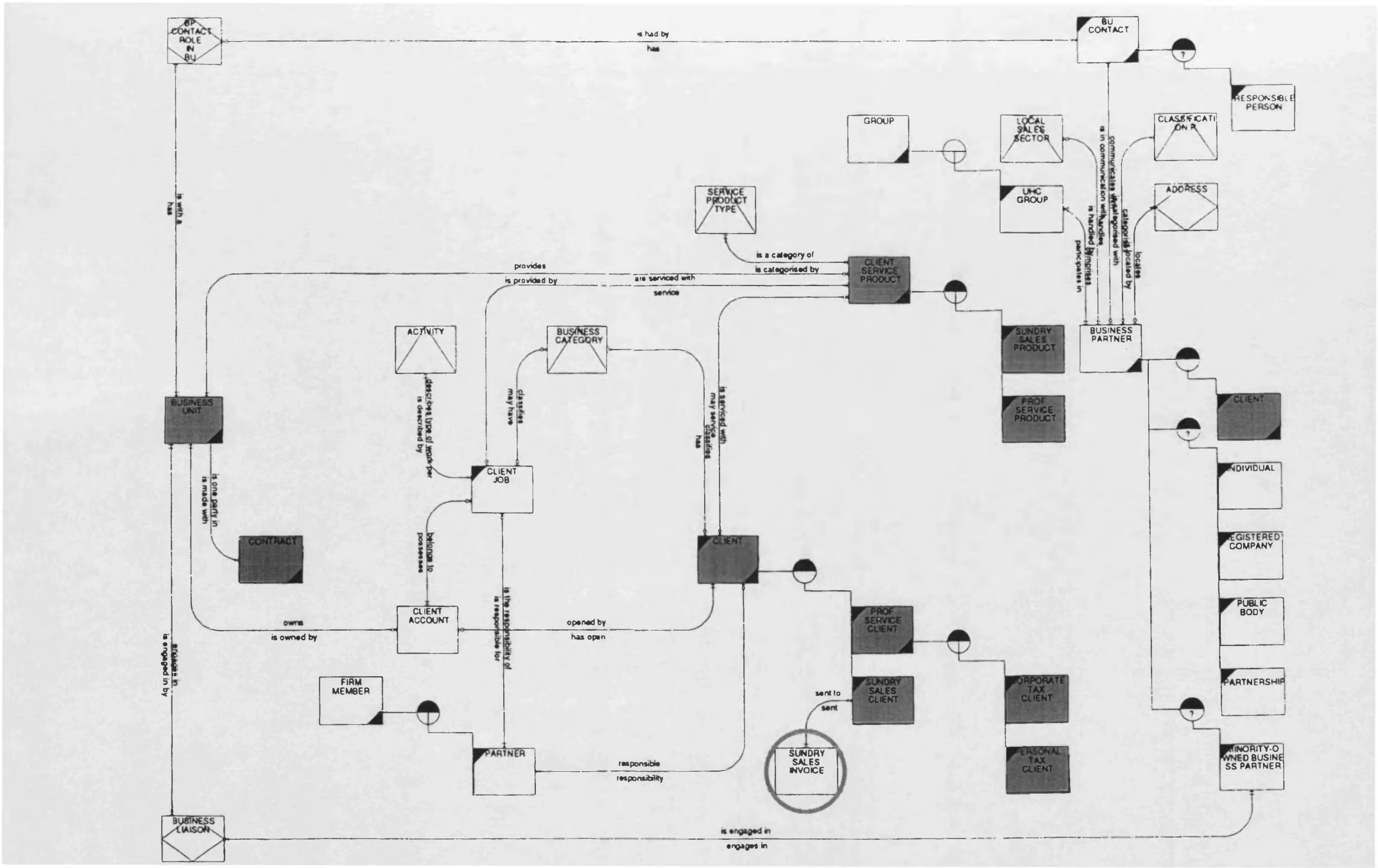
Figure 6.8 More details about connection between client and (client) service products

> ***Observation 8:*** *The natural clustering attribute of the semantic agent-based formalism contributes into the economy and readability of the ontology chart. It also augments the robustness of the model when it comes to the connections between different clusters.*
>
> ***Observation 9:*** *The semantic connections between two generic affordances from two different clusters can be attributed to the connections between their specifics.*
>
> ***Observation 10:*** *While adding new entities to an ER model can result in inconsistency, it seems that addition can hardly jeopardise the soundness of a semantic schema.*

The natural clustering attribute of the semantic analysis as a complexity reducing feature responds perfectly to the requirements for prototyping subsections of a large system discussed earlier in chapter 2 and addressed in chapter 4 which considered the findings of the exploratory case study. Clustering the system to the related subject areas will facilitate modular development of prototypes, the traceability of prototypes to their original requirements and the negotiation of meaning within the context of each subject. In the next page, there is a model with a highlighted area which shows the representation of the sundry sales invoice within the corporate data model. Referring to the definition from data dictionary:

Sundry Sales Invoice: A request for payment relating to the sale of sundry products.

The project manager agreed during our discussions that this entity represents not only an unimportant task at the operational level which does not need to be addressed at the corporate level, but also the way it appears in the model shows the inclusion of low level paperwork into the model. The task of invoicing at the operational level might be modelled in ontology chart, though not necessarily, as in figure 6.9.


BP CONTACT ROLE IN BU
is had by
has
BU CONTACT
RESPONSIBLE PERSON
GROUP
LOCAL SALES SECTOR
CLASSIFICATION
UHC GROUP
ADDRESS
SERVICE PRODUCT TYPE
is a category of
is categorised by
provides
is provided by
are serviced with
service
CLIENT SERVICE PRODUCT
is in communication with
handles
is handled by
participates in
categorised by
located
located by
SUNDRY SALES PRODUCT
PROF SERVICE PRODUCT
BUSINESS PARTNER
ACTIVITY
BUSINESS CATEGORY
describes
type of work per
is described by
classifies
may have
is serviced with
may service
classifies
has
BUSINESS UNIT
is with a
is one party in
is made with
CONTRACT
CLIENT JOB
belongs to
possesses
CLIENT
CLIENT
INDIVIDUAL
REGISTERED COMPANY
PUBLIC BODY
PARTNERSHIP
MINORITY-OWNED BUSINESS PARTNER
owns
is owned by
CLIENT ACCOUNT
is the responsibility of
is responsible for
opened by
has open
PROF SERVICE CLIENT
SUNDRY SALES CLIENT
CORPORATE TAX CLIENT
PERSONAL TAX CLIENT
FIRM MEMBER
PARTNER
sent to
sent
SUNDRY SALES INVOICE
responsible
responsibility
is engaged in by
engages in
BUSINESS LIAISON
is engaged in
engages in

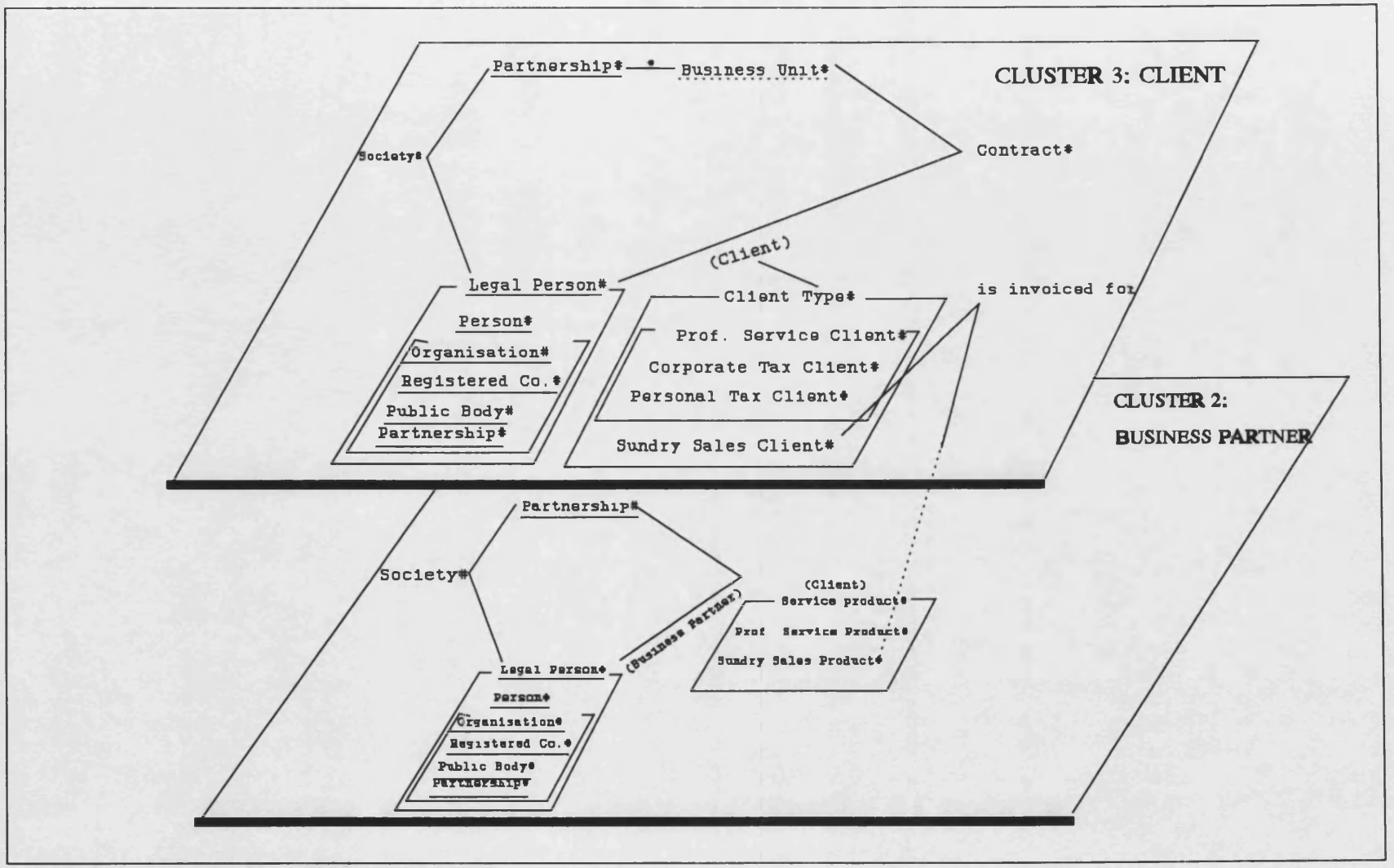


Figure 6.9 Invoicing the sundry sales client

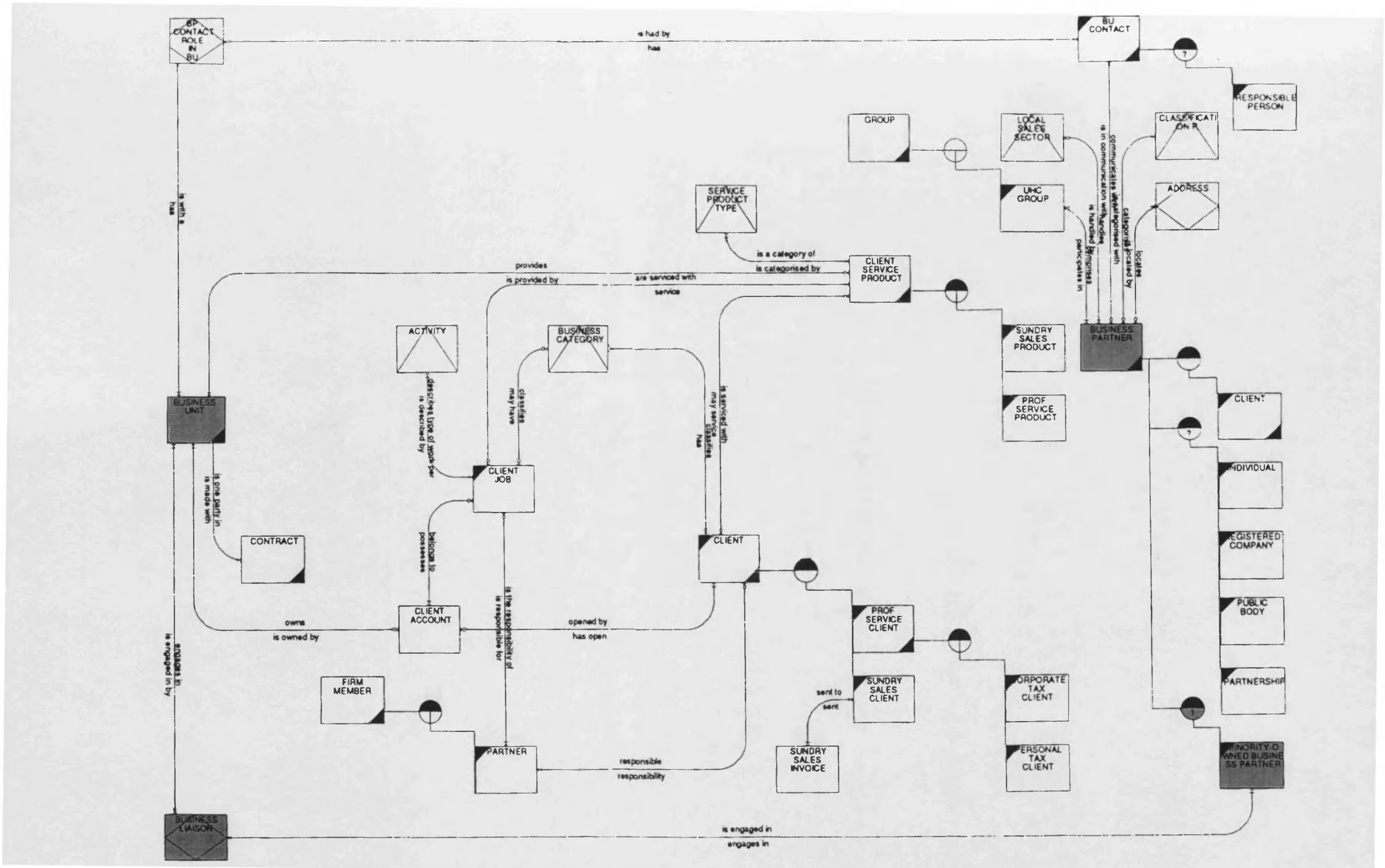
> ***Observation 11:*** *Since the ontology charts do not incorporate into the structure of the representation any normative rules or paper-handling activities, the effect can be to avoid hostages to fortune. If the corporate data model embodies business norms into its underlying data framework, when company decides to do things differently there will be a big overhead to pay in eradicating the now mistaken assumption.*

6.2.5 Minority owned business partner

Minority owned business partner is one of the concepts within the corporate data model under study that project senior analyst had a problem of how this entity should be represented in ER model. As shown in the next page model, this entity is treated as a subtype for business partner, which has also relationship to business liaison. The description of this entity is given as follows:

Minority Owned Business Partner: A party in a **Business Liaison** with the Firm where firm has an interest in promoting the business activities, because either directly or indirectly it results in the firm's profitability.

The concept of minority owned business partner in the ontology chart is represented in figure 6.10. This concept is a semantic outcome of the joint realisation of two concepts: business partnership and minority ownership. Minority ownership of an organisation by the partnership labels the company as minority owned and as the minority owned company participates as one of the partnership's business partners, this provides the notion of minority owned business partner. Although the way this concept is represented in the ontology chart seems to be a very straight forward, the fact is that the treatment of ontological constraints in semantic-agent based formalism indicates the pattern of behaviour and actions that needs to be realised before other concepts can be brought to life. This indication as a guiding force helps the analyst to maintain the semantic integrity of the model and to avoid doubtful representations such as appear in the next page model, where there is a question mark over the connection between business partner and minority owned business partner. The project manager agreed on the clarity of the semantic schema for this representation.


BP CONTACT ROLE IN BU
is had by
has
BU CONTACT
RESPONSIBLE PERSON
GROUP
LOCAL SALES SECTOR
CLASSIFICATION
UHC GROUP
ADDRESS
SERVICE PRODUCT TYPE
is a category of
is categorised by
provides
is provided by
are serviced with
service
CLIENT SERVICE PRODUCT
SUNDRY SALES PRODUCT
PROF SERVICE PRODUCT
BUSINESS PARTNER
ACTIVITY
BUSINESS CATEGORY
describes type of work done
is described by
classifies
may have
is serviced with
may service
classifies
has
BUSINESS UNIT
is a party in
is made with
CLIENT JOB
CONTRACT
CLIENT
belongs to
possesses
owns
is owned by
CLIENT ACCOUNT
is the responsibility of
is responsible for
opened by
has open
PROF SERVICE CLIENT
SUNDRY SALES CLIENT
CORPORATE TAX CLIENT
PERSONAL TAX CLIENT
FIRM MEMBER
PARTNER
responsible
responsibility
sent to
sent
SUNDRY SALES INVOICE
CLIENT
INDIVIDUAL
REGISTERED COMPANY
PUBLIC BODY
PARTNERSHIP
MINORITY-OWNED BUSINESS PARTNER
is engaged in by
engages in
BUSINESS LIAISON
is engaged in
engages in

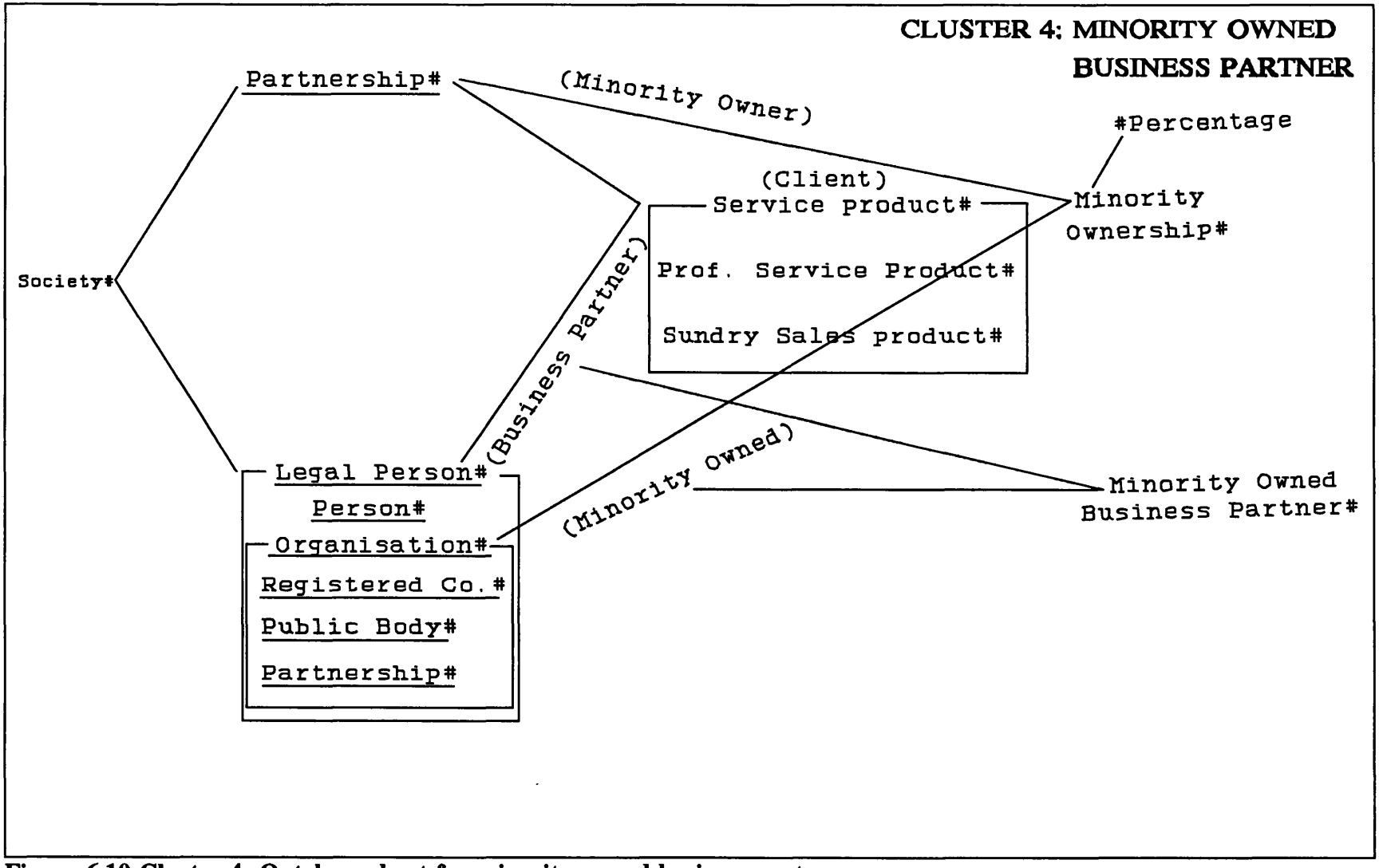


Figure 6.10 Cluster 4: Ontology chart for minority owned business partner

Observation 12: *Ontological dependency and the treatment of semantic modelling in ontology charting to be readable from left to right, provides a guiding force for analyst to maintain the semantic integrity of concepts. It shows the way that how realisation of one concept can lead to the next.*

6.2.6 Business liaison

The definition of business liaison in the data dictionary of the corporate data model is presented as follows:

Business Liaison: A description of firm's business interest in the relationship with the **Business Partner**.

As shown on the ontology chart on next page (figure 6.11), a connection between two clusters creates the notion of business liaison. The treatment of natural clustering again brings forward a robust and stable semantic schema which automatically places different concepts in their related cluster and then, when necessary, nicely connects them to each other to represent the new concepts.

6.2.7 Client job

Client job is part of the (client) service product which services a client. Client job which is described in form of activities is the direct responsibility of partner in his relationship with the client. The definitions of these entities are given in the data dictionary as follows:

Client Job: An auditable unit of work performed. Synonyms used in other parts of the Firm include: engagement, assignment or project(one-off). Time and expenses for a **Client Job** is recorded as **Charge Account**. It records the **Business Unit** partner and Manager responsible for the **Charge Account** and also the firm's categorisation of the **Charge Account**.

Activity: A classification of the type of work supplied by the firm.

Service Product Type: A category of professional service or sundry item on offer by the firm. This equates to the Time Accounting Activity Code.

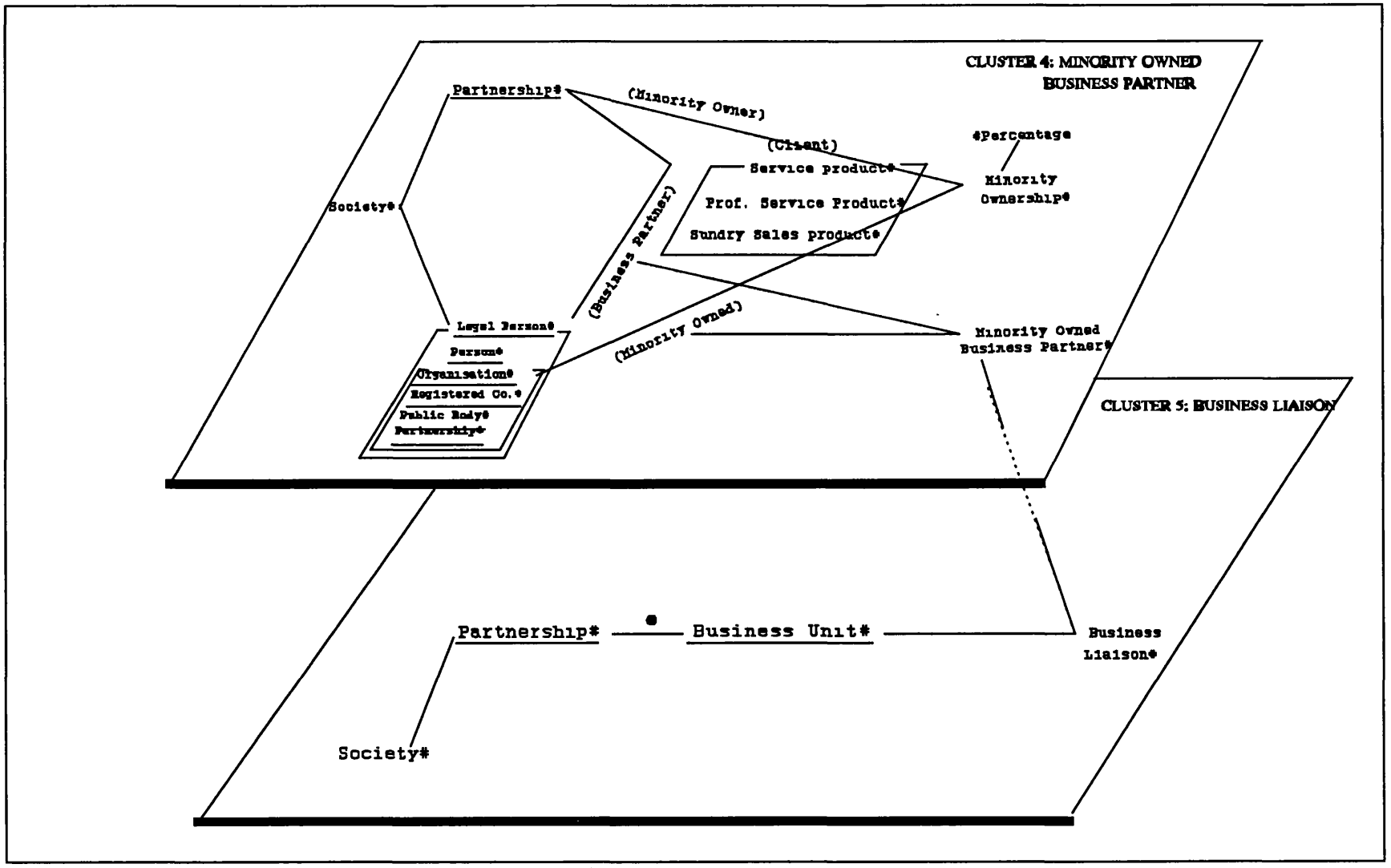


Figure 6.11 Cluster 5: Ontology chart for business liaison

The next page model highlights the participating entities for this cluster of semantic schema which is represented as more additions to cluster 2 (business partner) previously discussed in subsection 6.2.3. Therefore the following ontology chart represented in figure 6.12 provides more details of cluster 2 for business partner and client job.

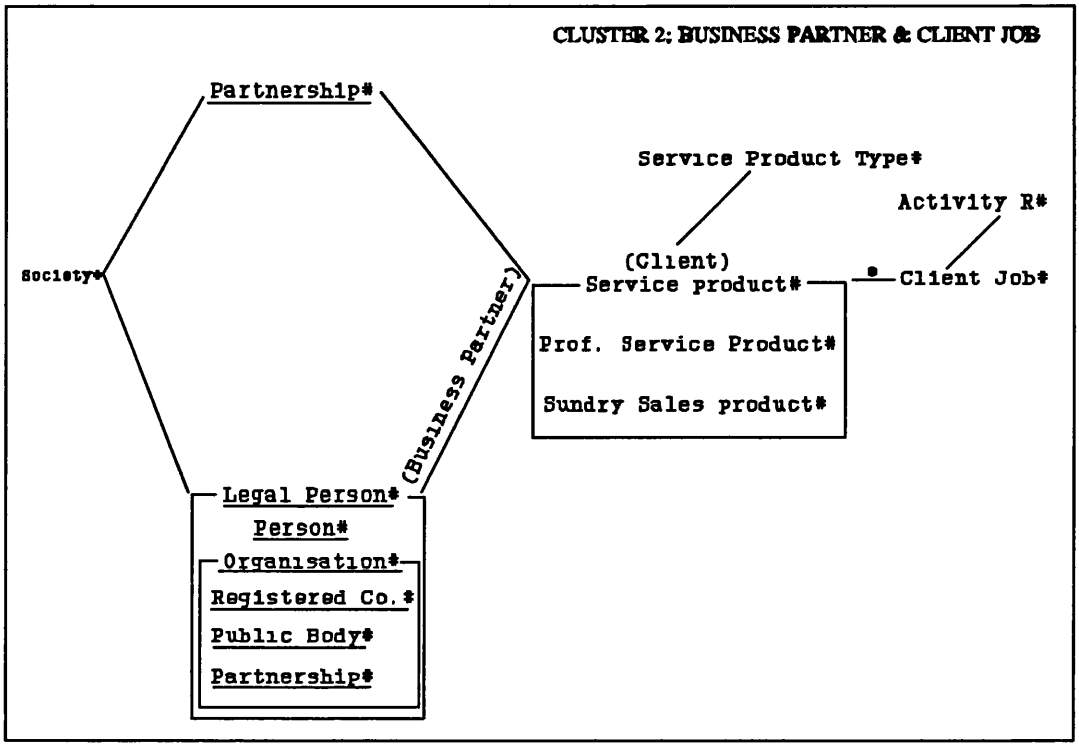
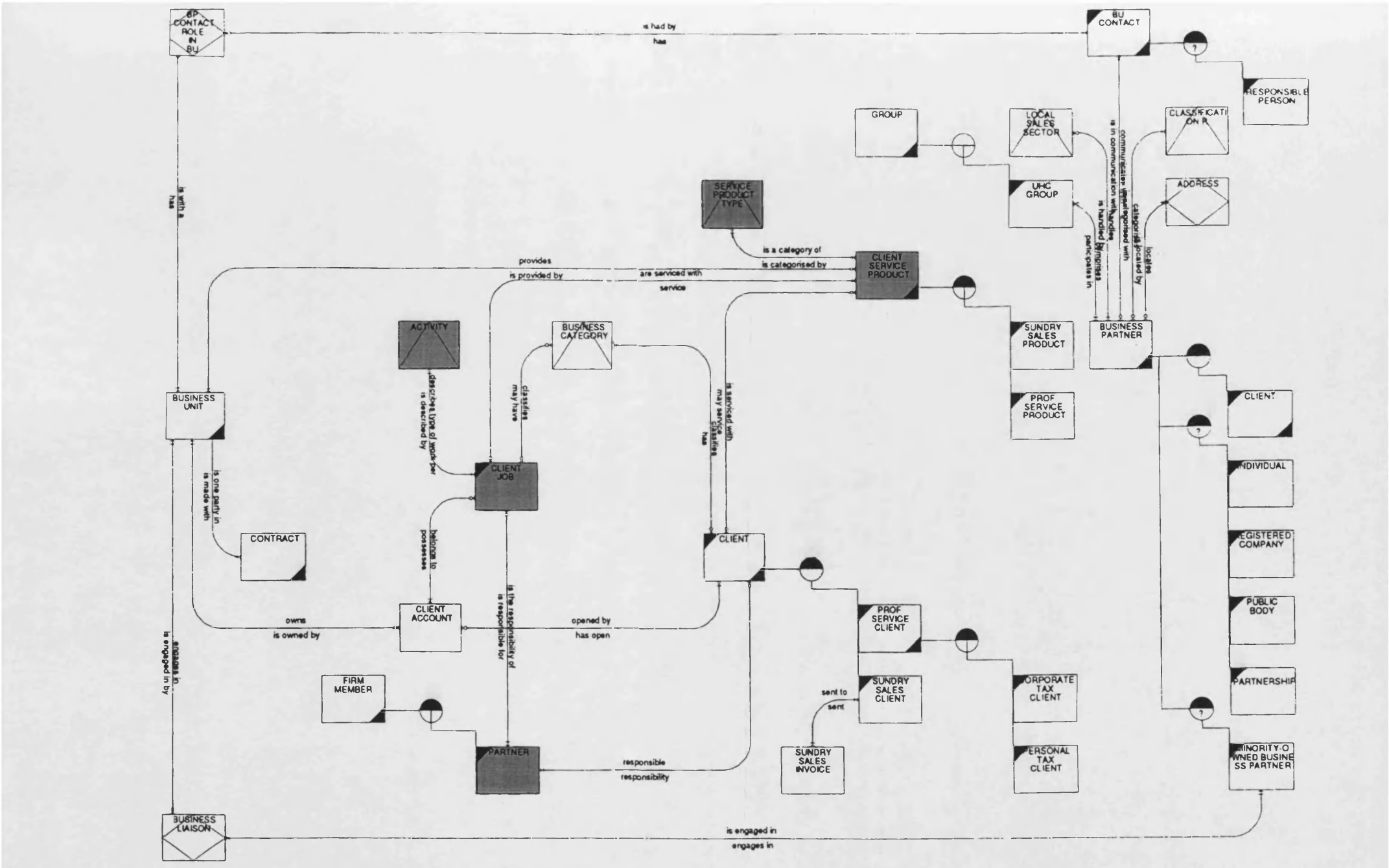


Figure 6.12 More details about business partner and client job (extension of cluster 2)

As shown in the above figure, the whole-part relationship in the semantic agent-based formalism represents a very rigid semantic notion. The corporate data model and even the description given in data dictionary do not specify whether client job is nothing other than part of services designed for a client.

> ***Observation 13:*** *Whole-part relationship as an ontological dependency: a part cannot exist without the whole, provides a useful mechanism to distinguish dependent concepts. Although in other modelling formalisms, such as object modelling technique, we can see a similar concept in form of aggregation of components, but it does not convey the ontological dependency of components on the aggregate and mainly conveys the idea of assembling physical things together.*


BP CONTACT ROLE IN BU
is had by
has
BU CONTACT
RESPONSIBLE PERSON
GROUP
LOCAL SALES SECTOR
CLASSIFICATION R
UHC GROUP
ADDRESS
SERVICE PRODUCT TYPE
is a category of
is categorised by
provides
is provided by
are serviced with
service
CLIENT SERVICE PRODUCT
is with a
has
communicates with/addresses
is in communication with/addresses
is handled by/mailed
participates in
categorised/located by
locates
ACTIVITY
BUSINESS CATEGORY
SUNDRY SALES PRODUCT
BUSINESS PARTNER
describes/ type of work for
is described by
classifies
may have
is serviced with
may service/classifies
has
PROF SERVICE PRODUCT
BUSINESS UNIT
CLIENT
CLIENT JOB
is one party in
is made with
CONTRACT
belongs to
possesses
CLIENT
INDIVIDUAL
REGISTERED COMPANY
PUBLIC BODY
owns
is owned by
CLIENT ACCOUNT
is the responsibility of
is responsible for
opened by
has open
PROF SERVICE CLIENT
PARTNERSHIP
FIRM MEMBER
sent to
sent
SUNDRY SALES CLIENT
CORPORATE TAX CLIENT
is employed in
employs
PARTNER
responsible
responsibility
SUNDRY SALES INVOICE
PERSONAL TAX CLIENT
MINORITY-OWNED BUSINESS PARTNER
BUSINESS LIAISON
is engaged in
engages in

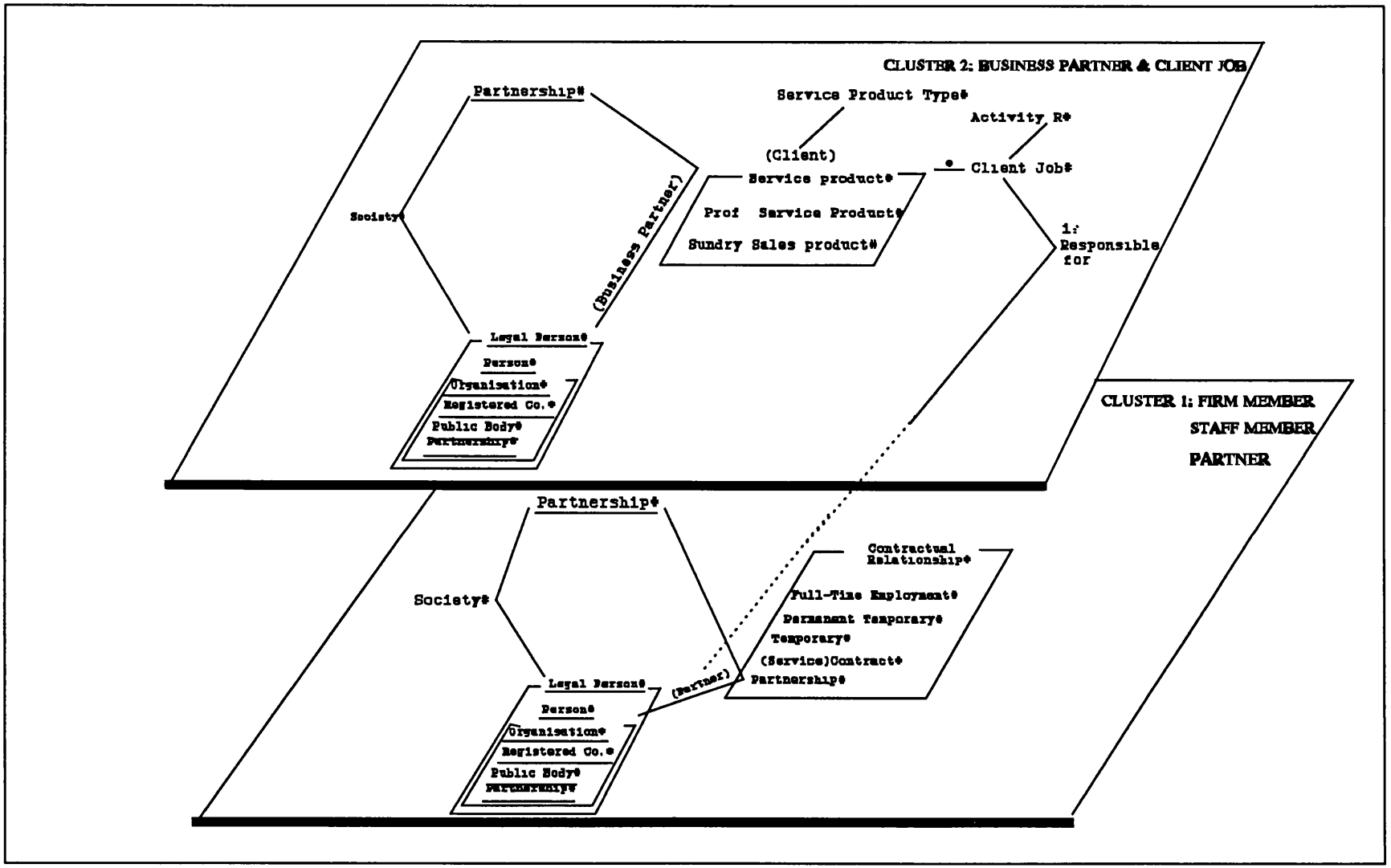


Figure 6.13 A complete ontology chart for cluster 2: business partner and client job

Figure 6.13 in previous page completes the ontology chart of cluster 2 for business partner and client job by representing the concept of responsibility for partner in the fulfilment of client job.

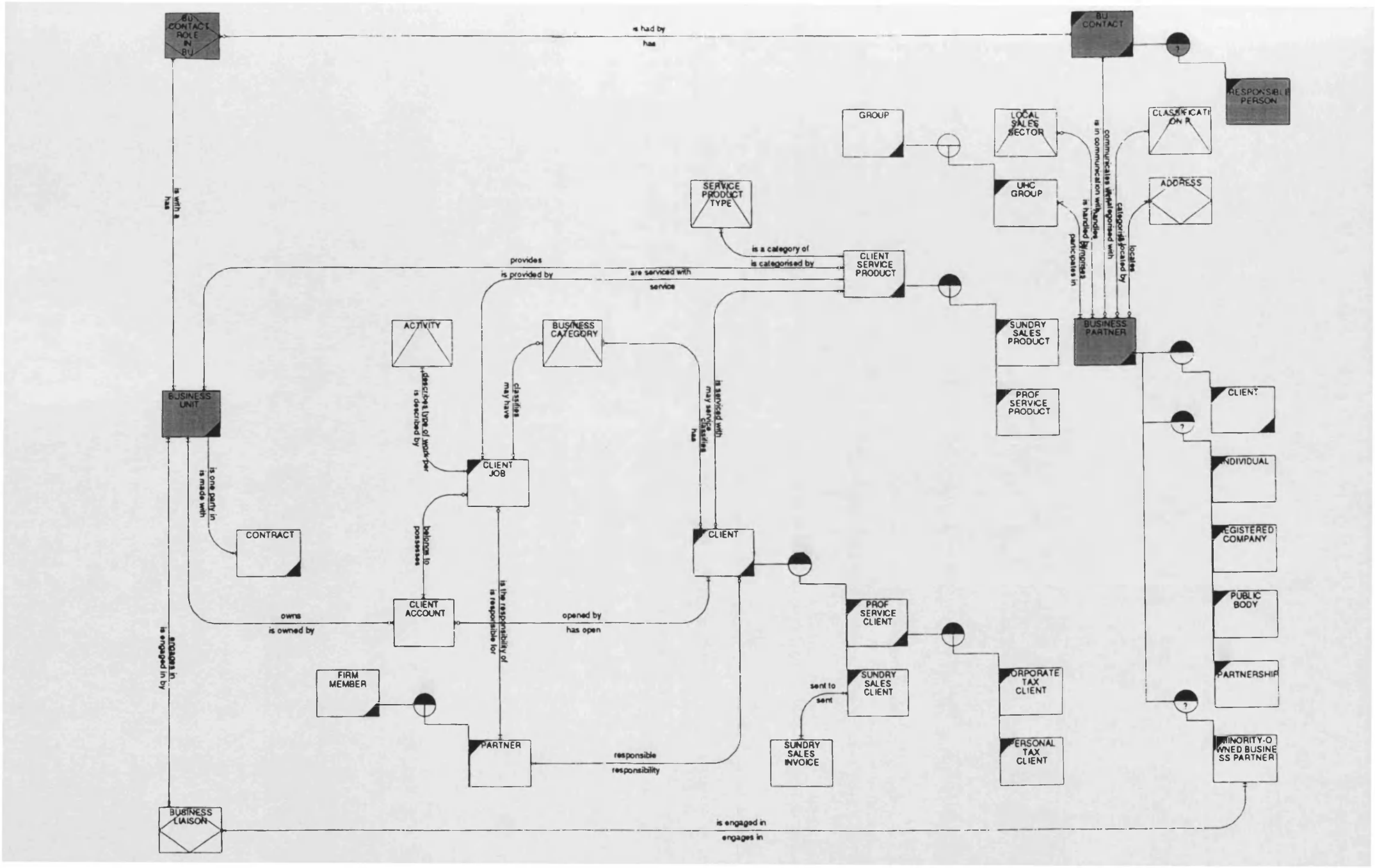
6.2.8 Business unit contact role

The model on the next page highlights that part of the corporate data model which is the subject of this subsection. This subsection models cluster 6 of the semantic schema as ontology chart for business unit contact role. It mainly focuses on the findings about the semantic treatment for different type of roles. Before discussing these types of roles, just a quick look at the dispersed highlighted entities in different corporate data models represented so far, brings any reader's attention to the highly scattered placement of entities within the model.

> ***Observation 14:*** *The pattern of placement of entities in the ER model (and possibly in other modelling formalisms) has no sense from a semantic viewpoint. It is arbitrary with no semantic constraints which makes the model very difficult to read and obtain to achieve understanding about it. In the semantic agent-based formalism the semantically correct place for each affordance is determined by its contribution to the creation of meaning. In this way better readability, natural clustering and manageable maintenance may be achieved.*

One of the most common defects in the semantic integrity of the corporate data model studied for this case is the treatment given to roles and relationships. In the semantic formalism we adopt there are three types of roles recognised which shows the power of the formalism for role clarification, discussed earlier in previous section:

> **Roles within relationship**
>
> A role can arise within a relationship such as parenthood (mother, child) or within ownership (property, owner). The important thing is that whatever is filling a role can also participate in other relationships and hence acquire


BU CONTACT ROLE IN BU
is had by
has
BU CONTACT
RESPONSIBLE PERSON
GROUP
LOCAL SALES SECTOR
CLASSIFICATION
UHC GROUP
ADDRESS
SERVICE PRODUCT TYPE
is a category of
is categorised by
CLIENT SERVICE PRODUCT
provides
is provided by
are serviced with
service
is with a
has
communicates with/handles
is in communication with/handles
is handled by/communicates with
participates in
categorised with
categorises
located by
locates
ACTIVITY
BUSINESS CATEGORY
SUNDRY SALES PRODUCT
BUSINESS PARTNER
PROF SERVICE PRODUCT
BUSINESS UNIT
describes/type of work performed by
is described by
classifies
may have
is serviced with
may service/classifies
has
CLIENT JOB
CLIENT
INDIVIDUAL
is one party in
is made with
CONTRACT
REGISTERED COMPANY
belongs to
possesses
PUBLIC BODY
owns
is owned by
CLIENT ACCOUNT
opened by
has open
is the responsibility of
is responsible for
PROF SERVICE CLIENT
PARTNERSHIP
FIRM MEMBER
sent to
sent
SUNDRY SALES CLIENT
CORPORATE TAX CLIENT
PARTNER
responsible
responsibility
SUNDRY SALES INVOICE
PERSONAL TAX CLIENT
MINORITY-OWNED BUSINESS PARTNER
is engaged in by
engages in
BUSINESS LIAISON
is engaged in
engages in

other roles. This is a very powerful and efficient use of language referring to someone as a divorced bankruptee tells a whole story about a person very succinctly. But it would be a mistake to record these persons in virtue solely of their roles. Hence recording a legal person as a client begs the question of under what circumstances it commences or ceases to be a client.

Roles as part of structure

Other roles include positions within an organisational structure. These are also referred to as offices, posts or positions. They are part of the organisational framework and may survive longer than any particular incumbency. Each organisation consists of a number of positions (roles) which might be filled by different persons in different periods of time. There are problems with coping with the changes in organisational structure as well as with recording which person is filling the position over time.

Roles as attribute

A third possibility is when a person acquire a role as part of his definition. Pavarotti is a singer and will be referred to in this way even when his singing career ends.

The first two role types are the most important for business. So far we have had number of examples for the first type of role, eg. client, business partner, .. In all of those examples relationships are presented as having paired antecedent principles, and roles arise during the existence of the relationship, and also cease when the relationship ceases. Here for business unit contact role we are dealing with the second type of roles mentioned earlier: roles as positions or offices within the firm's organisational structure. It is apparent from the following data dictionary definitions that business unit contact role is a position within the business unit as part of the partnership.

BU contact role in BU: Describes the participation of the **BU Contact** in the **Business Unit.**

Business Unit Contact: A Firm Member who represents the **Business Unit** in its dealings with an **Organisation.** In a client service **Business Unit** may be a subtype of **Firm Member role in Business**

Unit.

Responsible Person: A Firm Member with responsibility for a particular area.

Figure 6.14 represents the ontology chart for business unit contact role as a specific role within the spectrum of positions in business unit.

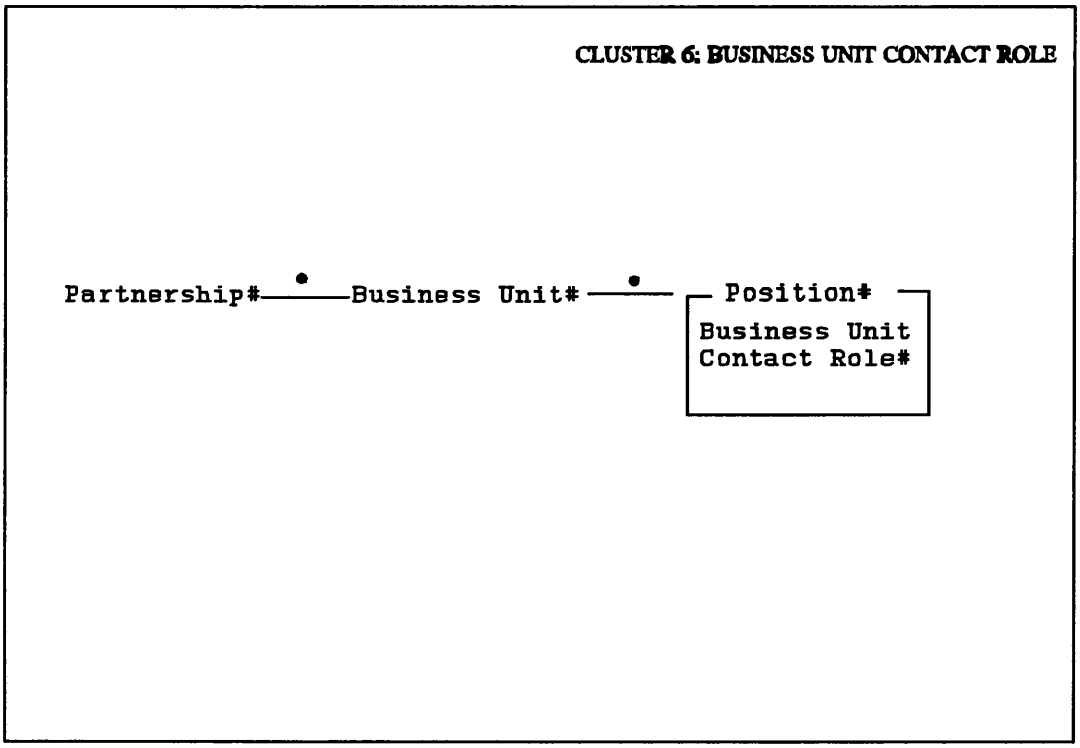


Figure 6.14 cluster 6: Ontology chart for business unit contact role

The notion of responsibility for communication between a business partner and a responsible person within the firm comes hand in hand with the notion of incumbency by the person in a position which is the target of communication by business partner. So semantically we need to distinguish between a person as incumbent or holder of a position and the position itself. This detection is naturally handled in semantic analysis as shown in the semantic schema on the next page (figure 6.15) for this case:

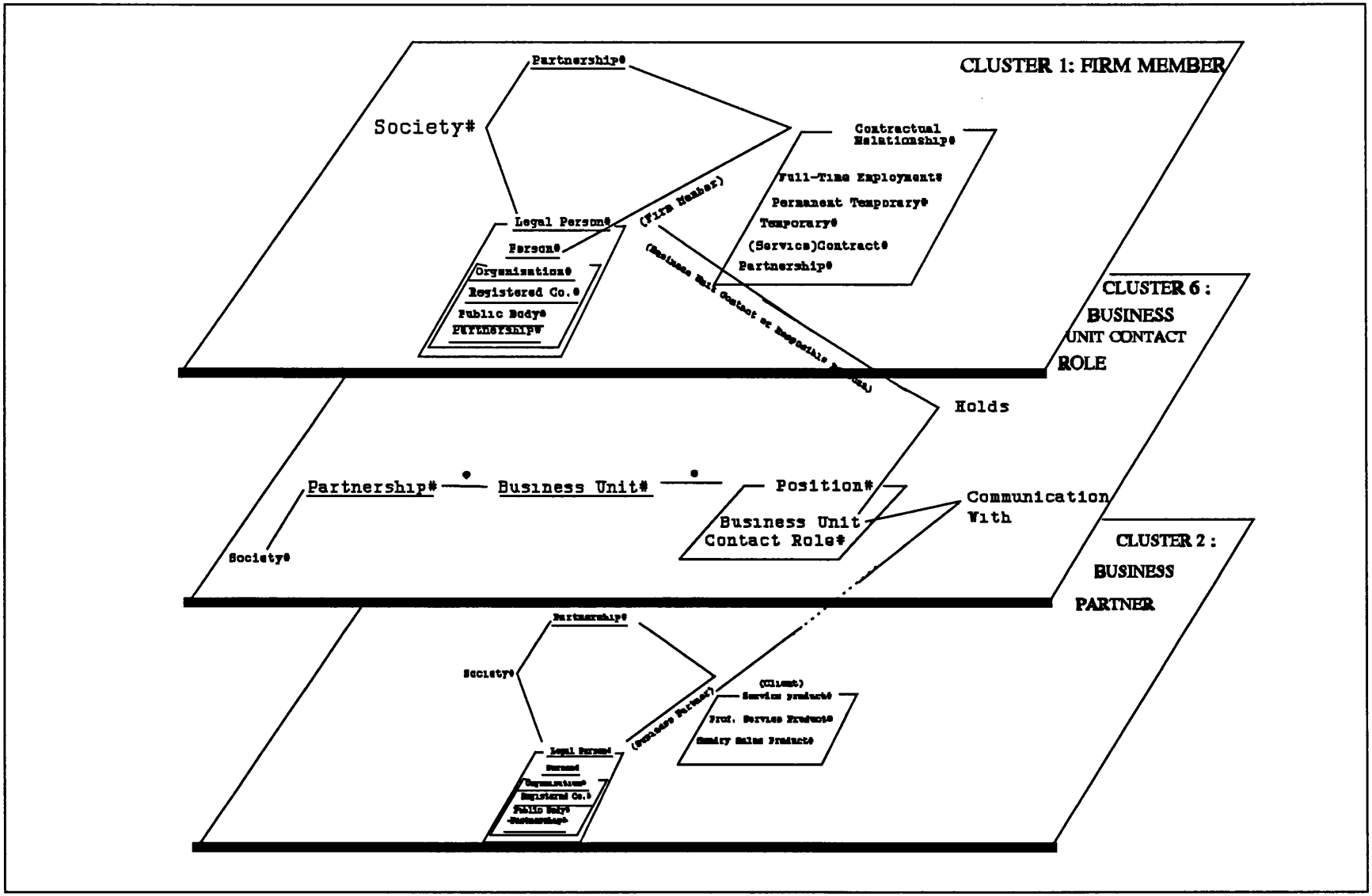


Figure 6.15 More details about business unit contact role

The discussion so far regarding the roles and relationships and the unique handling of them in the semantic agent-based formalism leads to observation 15:

> ***Observation 15:*** *The semantic agent-based formalism adopted in ontology charting handles roles and relationships naturally.*

As the figure 6.15 in the previous page demonstrates, a firm member who holds the position of business unit contact role is labelled as business unit contact or responsible person. During the discussions with project members, it appears that the both terms: business unit contact (person) and responsible person are indeed two names for one thing. Inclusion of these names in different ways, one as sub-type of the other was mainly because of inability of ER modelling to handle the placement of responsibility where it seems to be absolutely critical.

> ***Observation 16:*** *Concepts which have the same name do not carry the same significance throughout the company. Equally two concepts that have the same meaning may be known by different names, or aliases. In the end the only way of knowing is to look at how these concepts are realised in practice. And practice may reveal that there are indeed totally different significations running alongside each other.*

The power of ontology charts in forcing participants, especially persons from the law and regulation division and the project manager, to discuss the meaning of terms was a significant observation from the presentations of the ontology charts as the results of our semantic analysis. The discussions stimulated by semantic schema specifically in some occasions like minority owned business partner or this current cluster about business unit contact role showed us how the semantic analysis technique can really trigger the negotiation of meanings where semantic ambiguities are involved.

> ***Observation 17:*** *Semantic modelling technique has the ability to trigger discussions and to pave the way for conceptual*

training of participants during the course of analysis. By demanding semantic integrity in every aspect of ontology charts, there is a driving force for achieving a semantic consensus on the realisation of different concepts in practice.

It is interesting to consider the treatment of communication between business partner and business unit contact role in the semantic schema represented in figure 6.15. In the original ER model the communication from business partner is directed to business unit contact as the incumbent of the business unit contact role. But the semantic constraints of ontology charting avoids such a dependency taking place. In reality the business partner communicates with the position responsible to him. If on any occasion, even for a very short period of time, there were nobody to hold the position of business unit contact role, we cannot expect that the whole communication between the firm and its business partner collapses. Obviously from the business partner point of view, he communicates with the position and therefore with whoever is responsible to cover that post. This concept is demonstrated in the figure 6.15 when connecting cluster 2 to cluster 6.

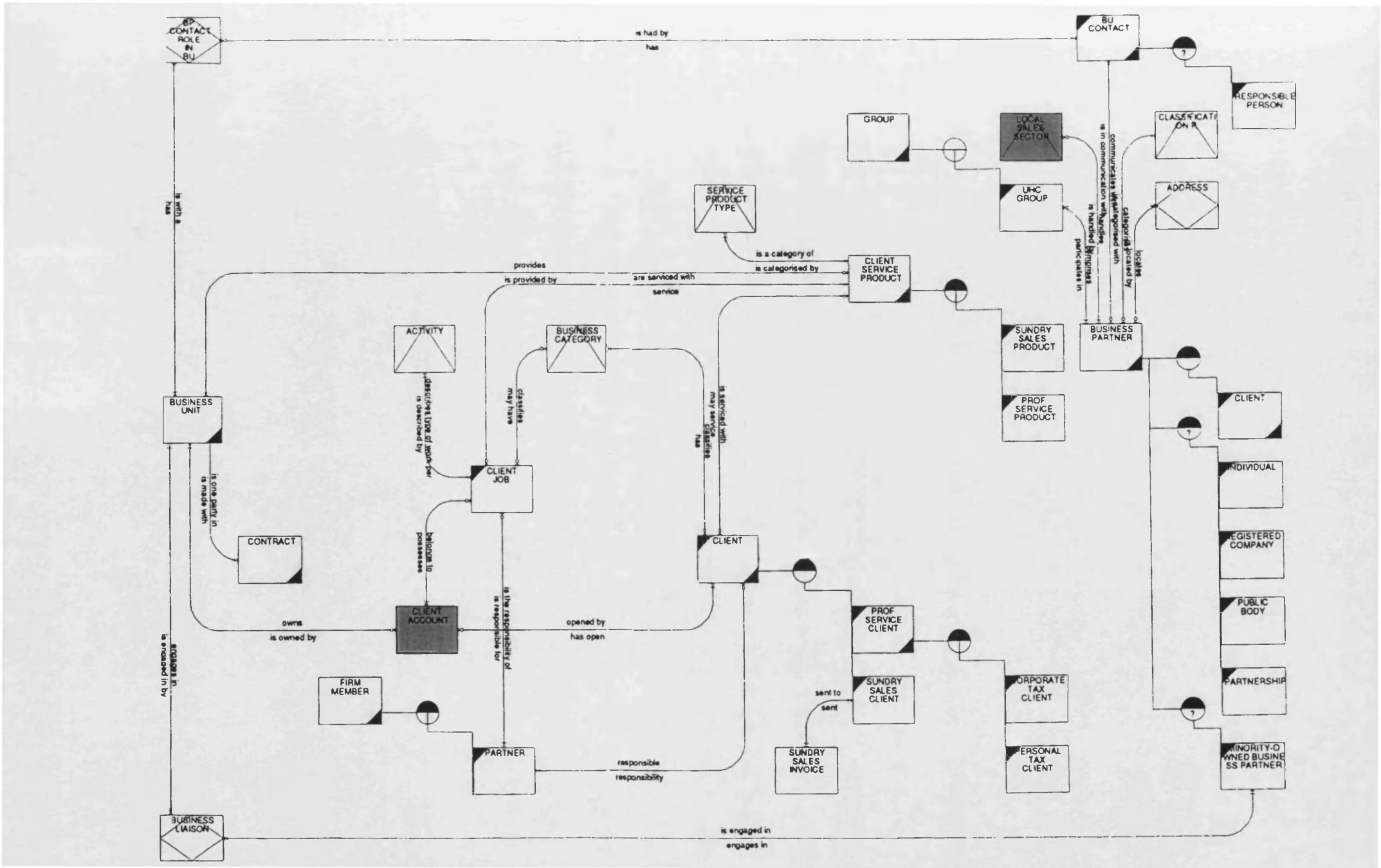
Observation 18: *The semantic constraints employed in the semantic analysis technique and the logic of ontological dependency prevent imposing any artificial dependency between different affordances while it appears that ER modelling is highly susceptible to this weakness of modelling representation.*

6.2.9 Client account

The model in the next page shows how client account is related to other entities in the original corporate data model. Again the definitions from data dictionary are:

Client Account: A record of the financial relationship between the Firm and a **Client**. It may comprise a number of individual **Charge Accounts**.

Local Sales Sector: A categorisation of **Clients** at the **Charge Account/Client Account** level denoting the market sector to which the Firm/Industry Leader wishes to assign them.


BP CONTACT ROLE IN BU
is had by
has
BU CONTACT
RESPONSIBLE PERSON
GROUP
LOCAL SALES SECTOR
CLASSIFICATION
UHC GROUP
ADDRESS
SERVICE PRODUCT TYPE
is a category of
is categorised by
provides
is provided by
are serviced with
service
CLIENT SERVICE PRODUCT
SUNDRY SALES PRODUCT
PROF SERVICE PRODUCT
BUSINESS PARTNER
ACTIVITY
BUSINESS CATEGORY
BUSINESS UNIT
CLIENT JOB
CONTRACT
CLIENT
CLIENT ACCOUNT
owns
is owned by
opened by
has open
PROF SERVICE CLIENT
SUNDRY SALES CLIENT
CORPORATE TAX CLIENT
PERSONAL TAX CLIENT
FIRM MEMBER
PARTNER
responsible
responsibility
SUNDRY SALES INVOICE
sent to
sent
INDIVIDUAL
REGISTERED COMPANY
PUBLIC BODY
PARTNERSHIP
MINORITY-OWNED BUSINESS PARTNER
BUSINESS LIAISON
is engaged in
engages in

Client account is the ontological dependent of the contract between client and business unit of firm. Charge accounts are also parts of the client account. The following figure (figure 6.16) shows the ontology chart for this cluster.

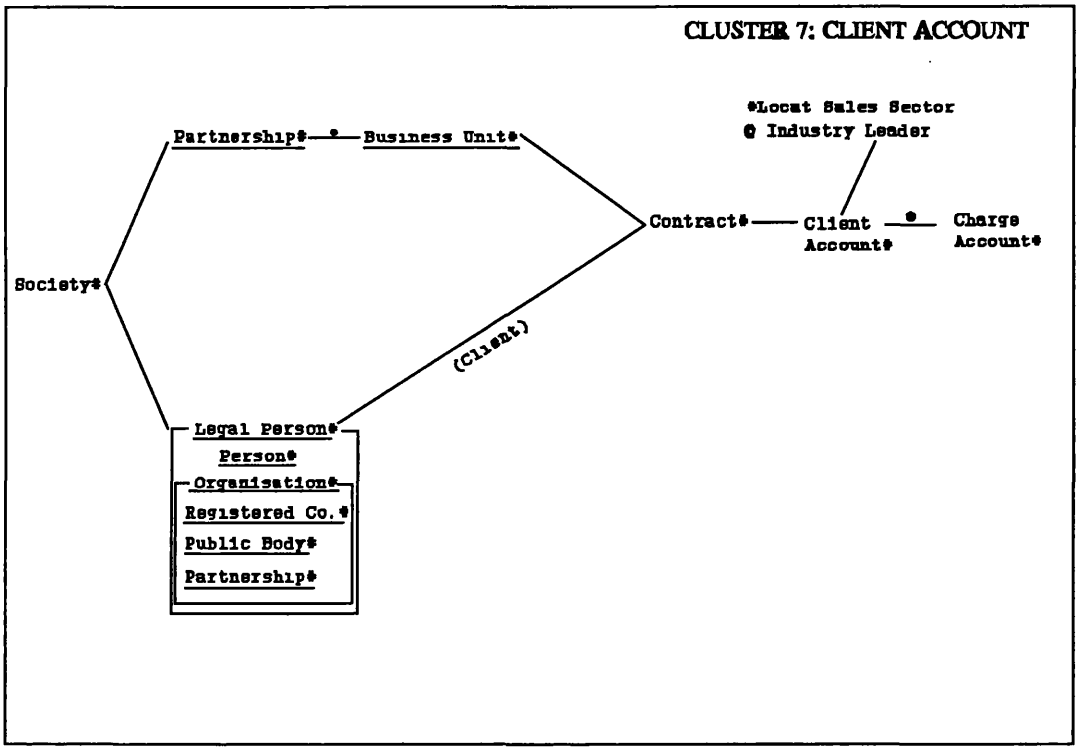


Figure 6.16 Cluster 7: Ontology chart for client account

During discussions with the project senior analyst , it became apparent that even the creation of client account is triggered by the realisation of a contract in the firms' financial division. The dependency of client account on client is also naturally maintained by the ontological dependency of contract on client. In the original corporate data model represented in previous page, in order to establish relationship in database system between client and his client account, the analyst has chosen to define an artificial relationship as client account **opened by** client out of necessity, while in practice no such action takes place by client.

> ***Observation 19:*** *The semantic agent-based formalism maintains the relationships between affordances as long as semantically they are relevant. It seems difficult to impose irrelevant ontological relationships for the sake of physical database requirements for example. This aspect of ontology charting*

might lead to yet higher levels of modelling consistency, robustness and opportunity for analysis reuse.

Again, during discussions and also from definitions in the data dictionary, it emerged that local sales sector is a determiner for client account and the authority for this determination belongs with industry leader for each sector. The relationship defined as local sales sector **handles** business partner is not consistent with the actual practice in the workplace. This will complete the ontology chart for this cluster, explaining the whole semantic model for client account.

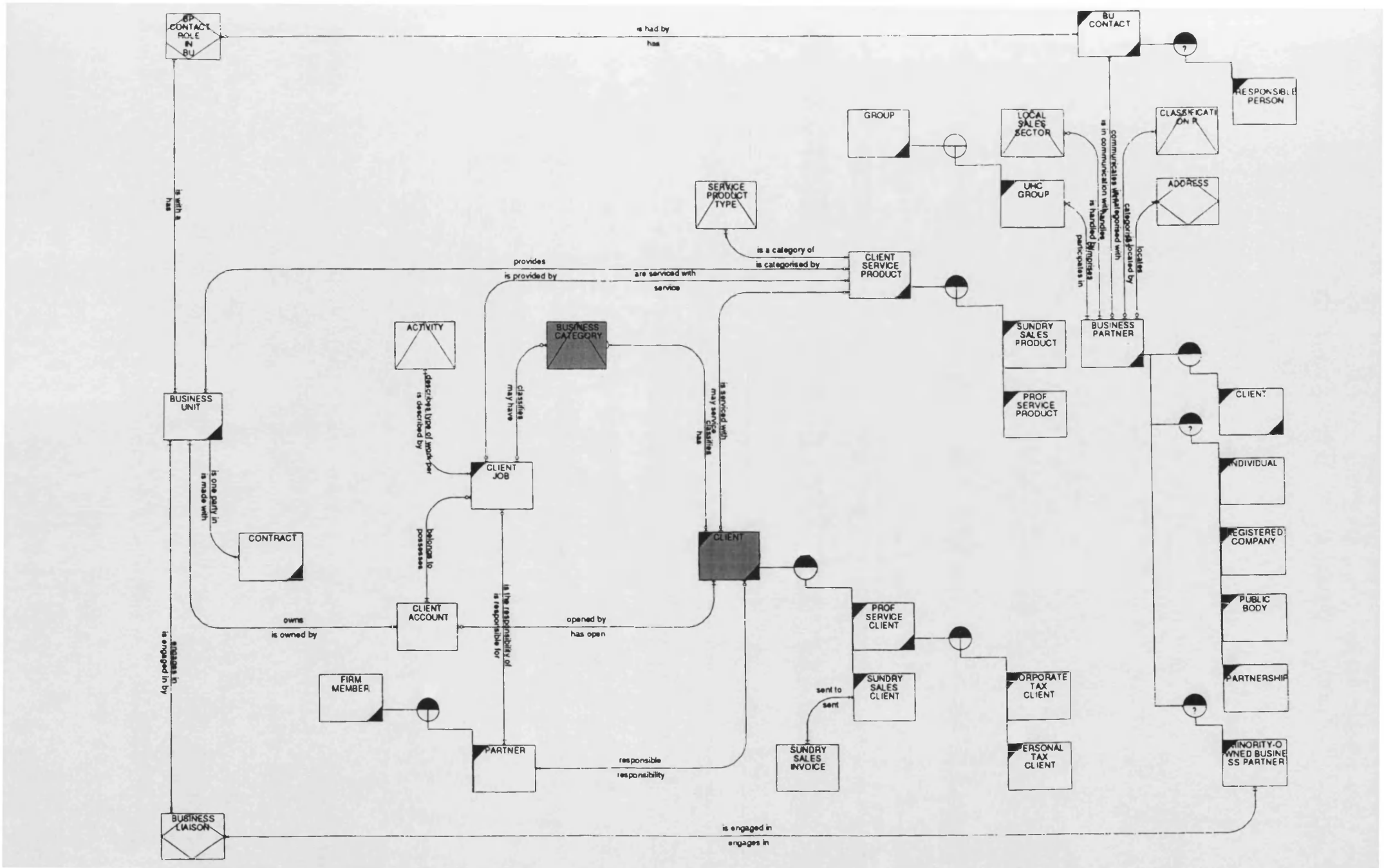
Observation 20: *Each cluster in the semantic schema naturally shapes itself. This will help in software development modularity and easier maintenance of software. It seems also to offer the possibility to design security structures based on natural clustering of the model, instead of artificial assignment of access codes to users.*

6.2.10 Business category

The definition for business category based on data dictionary is as follows:

Business Category: A Business Category denotes which of the Firm's specialist groups (SIG's) may have an interest in a **Client** or a **Charge Account**. It may be either an industry sector such as Banking or Insurance , or a specialist niche such as High-Tech, Pensions or Privatisation.

The above definitions lead to understanding that special interest groups are part of the firm and business category is a determiner for each group with two specific category: industrial sector and specialist niche. The relationship between this type of categorisation of clients of the firm is established through the interests of special interest groups from business category cluster (cluster 8) to client cluster (cluster 2) as represented in figure 6.17. The following pages show the original ER model for business category and the semantic schema for that part. The senior analyst and project manager mentioned in number of occasions that the natural emergence of the relationship between clusters is a useful feature in enhancing their understanding.


BP CONTACT ROLE IN BU
is had by
has
BU CONTACT
RESPONSIBLE PERSON
GROUP
LOCAL SALES SECTOR
CLASSIFICATION R
UHC GROUP
ADDRESS
is with a
has
SERVICE PRODUCT TYPE
is a category of
is categorised by
provides
is provided by
are serviced with
service
CLIENT SERVICE PRODUCT
SUNDRY SALES PRODUCT
PROF SERVICE PRODUCT
BUSINESS PARTNER
ACTIVITY
BUSINESS CATEGORY
BUSINESS UNIT
describes type of work per
is described by
classifies
may have
is serviced with
may service/classifies
has
CLIENT
INDIVIDUAL
CLIENT JOB
is one party in
is made with
CONTRACT
belongs to
possesses
REGISTERED COMPANY
PUBLIC BODY
owns
is owned by
CLIENT ACCOUNT
is the responsibility of
is responsible for
opened by
has open
PROF SERVICE CLIENT
PARTNERSHIP
FIRM MEMBER
SUNDRY SALES CLIENT
sent to
sent
CORPORATE TAX CLIENT
PARTNER
responsible
responsibility
SUNDRY SALES INVOICE
PERSONAL TAX CLIENT
MINORITY-OWNED BUSINESS PARTNER
is engaged in by
engages in
BUSINESS LIAISON
is engaged in
engages in

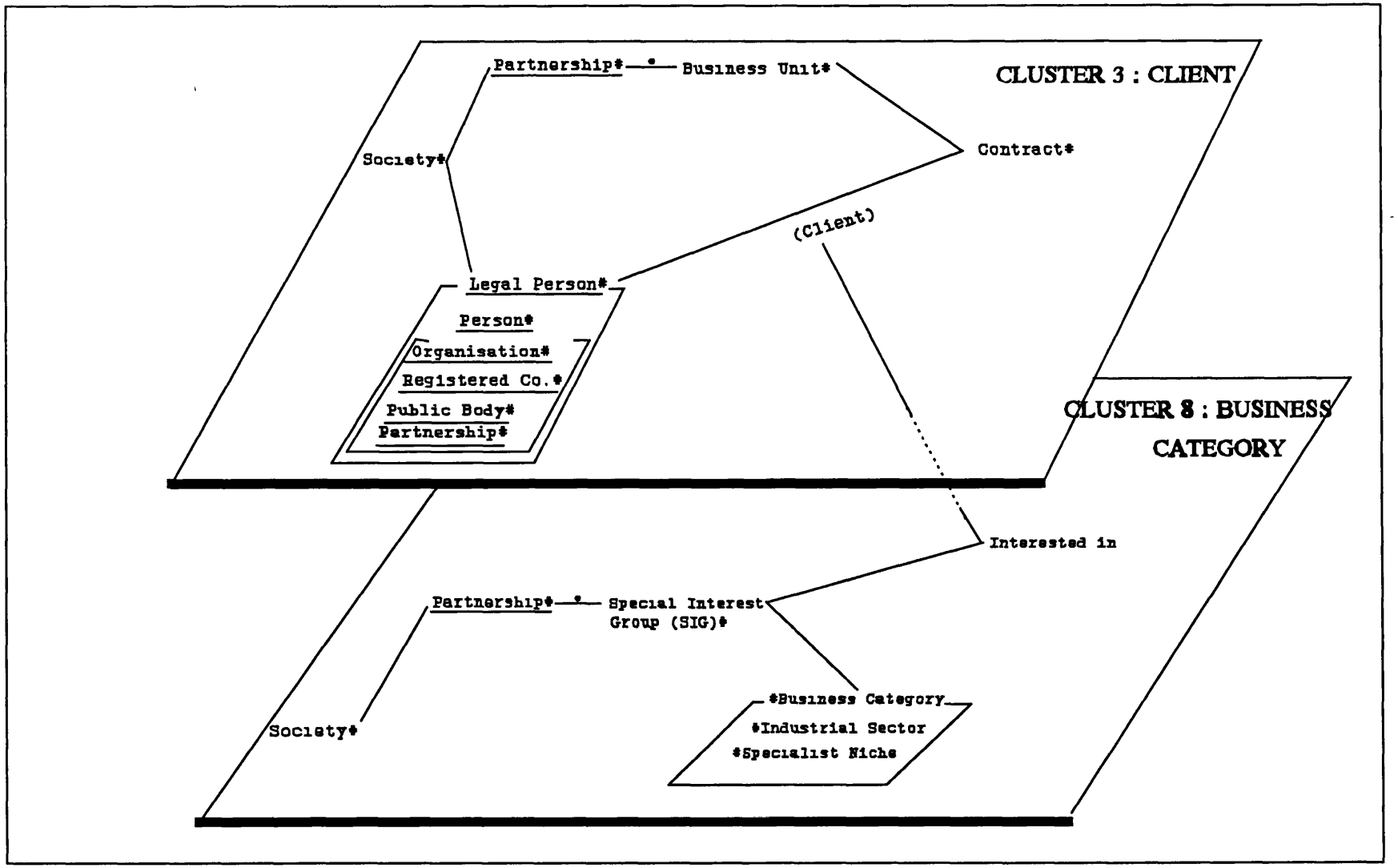


Figure 6.17 Cluster 8: Ontology chart for business category

> ***Observation 21:*** *The development of analysis appears to expand the understanding of the whole work practice. With each new cluster or new relationship between previously defined clusters, a new dimension of the corporate activities is unfolded and new levels of conceptualisation are captured. In other analysis approaches like the one employed in the creation of the corporate data model for this case, when analysis progresses the complexity of the model dramatically increases while traceability of previous analyses decreases.*

The resulting complicated ER model is a case in point for the above claim. All the persons involved in this case from the company agreed with this aspect of semantic analysis. The enhancement of understanding can facilitate the conceptual training of users and designers during the unfreezing stage of the development process and also supports the institutionalisation of changes in norms and routine activities of the organisation. The clear understanding with agreement on semantics of the workplace language bridges the semantic gap between parties involved in design and sets a unified frame of reference in unfreezing stage of development process before any commitment to further change development. This issue is explored from the previous case study reported in chapter 4 and also studied by proponents of planned change model mentioned earlier in previous section. The level of discussion, among all the persons involved in this case about ambiguities demonstrated the power of semantic analysis in resolving such ambiguities and complexities of design and in supporting the addition of knowledge. This might be claimed to show support for human interaction and knowledge transfer in the workplace. This strongly supports the development of design understanding.

> ***Observation 22:*** *The semantic analysis technique is congruent with requirements of unfreezing stage of change process discussed earlier in this thesis regarding conceptual training, role clarification and responsibility negotiation.*

> ***Observation 23:*** *Semantic analysis needs a fresh perspective on to the world under investigation, and hence is difficult to begin, especially for people who are used to seeing the world as a complex of given entities. But once analyst has started the analysis, it appears to be self-driven. It guides the analyst to creation of affordances to capture new concepts existing in practice. It has intrinsic features of self-creation and semantic self-regulation. It is an autopoietic analysis technique.*

The problem addressed in the above observation was the most important obstacle in starting the case and specially working with senior analyst during the development of semantic schemata. But the reward of completing the case compensated this difficulty and resulted in a better justification of the technique we applied in our proposal for a new method in evolutionary information systems development.

6.3 Conclusion

The conclusion of this chapter marks the end of the third stage of the research approach designed for this thesis. The empirical study discussed in the previous section provided a set of observations regarding various aspects of the semantic analysis technique. On a number of occasions, the technique showed its harmony with the needs of the **vision cycle** and the **fusion cycle** of the development method proposed in chapter 5. The semantic analysis technique, by focusing on the usage of language in the workplace, is a useful tool for bridging the semantic gap among users. It is shown by the findings of the explanatory case study that this technique might reduce semantic ambiguities and possibly lower the cost of negotiation in the unfreezing stage of implementation process. By highlighting semantic ambiguities, this technique will require responsible agents to seek for consensus in semantic disagreements before any commitment to development of prototype systems. These characteristics of semantic analysis accord well with the requirements of applying the planned change model studied by number of researchers and introduced as secondary materials in section 6.1.

One important feature of the semantic analysis technique and its harmony with the unfreezing stage of the planned change model is the recognition of an alternative perspective in the inquiry process by which the proposed method collects information for analysis and design through root meanings and conceptual models (see observations 6, 7, 15, 16, 17, 21 and 22). This focuses on improving the collective understanding of the problem situation.

The natural clustering feature of the technique also grants the prospects of analysis reuse and semantic-oriented integrity of user requirements through reducing complexity in the development of an enterprise information system (see observations 8, 9, 10, 14, 19 and 20). But this feature is also criticized for producing bulky documentation.

The findings of the explanatory case study show that the proposed method is most closely aligned with the "integrationist" dimension in viewing information systems development process (for example see observations 1, 12 and 23). It seems to favour coordination and integration. The search for a consensus during problem formulation involving multiple perspectives through the use of semantic analysis (and also prototyping) is clearly integrationist. Yet here can also be an essential weakness. We can be critical of the method's failure to reflect on whether it can be misused in realizing the goals of one group in the organisation at the expense of another. The method does not attempt to analyze and mitigate potential distortions and communication barriers while seeking a consensus on the creation of new meanings. Hence the cognitive basis of the semantic models may be flawed by an undetected bias. This also leads to ignoring the nature and influence of organisational power and it fails to be sensitive to the issue whether the new system will strengthen emancipation of all organisational participants or continued domination.

CHAPTER 7

Conclusions and recommendations

Chapter overview

This is the closing chapter of this thesis. Section 7.1 summarizes the research undertaken in the thesis. It also reviews the strengths and weaknesses of the proposed method in evolutionary development. Section 7.2 prefaces four avenues in the future research to extend this thesis.

7.1 Research summary

The research reported in this thesis focuses on planned organisational change and semiotic theory for use in evolutionary prototyping in information systems development. It adapts a subjective/argumentative research approach emphasizing socio-technical interpretation over the entire information systems development process.

In this thesis the basic features of information system design problems have been outlined as complexity and uncertainty, compounded by change in business environments. It is claimed that an evolutionary prototyping approach for eliciting user requirements might improve those difficulties in practice. In this thesis we have investigated this claim and tried to provide insights into difficulties of evolutionary development through conceptual and empirical analyses. More specifically, we have sought to meet the following two major goals:

1) to trace systematically the difficulties in managing evolutionary prototyping in an information systems development project back to a set of beliefs about its domain of change; and

2) to point out that evolutionary prototyping approach cannot be reduced to technological fixes.

The first goal was addressed by demonstrating that a wide range of issues is associated with managerial and social aspects of evolutionary prototyping. An exploratory case study, conducted at a large car manufacturer company, led us to perceive the inherent complexity of social change which is associated with information systems development. The second goal was addressed by pointing out that there is inherent complexity in the social condition and business change environment of systems development which escapes technological solutions; indeed, such complexity is often amplified through such technological solutions.

A deeper analysis of an evolutionary prototyping approach to information systems has been presented in terms of the lack of a development management process and the lack of a socially-based conceptual model to support conceptual training of all users before commencing any prototype development.

A planned change model consisting of a vision cycle (or unfreezing), an action cycle (or moving), and a fusion cycle (or unfreezing) was suggested to improve the management of the evolutionary process. Also, a semantic analysis technique from the perspective of semiotic theory was advanced to support enhancement and legitimation of any changes in user requirements during unfreezing and refreezing stages of the development process. On the basis of the analysis technique, this thesis has proposed the creation of a new perspective on evolutionary development for eliciting user requirements and performing design tasks guided by the planned change model.

The proposed perspective is supported by a theoretical framework consisting of planned organisational change theory and semiotic theory. The theoretical framework highlights the major difference between the proposed perspective and the prototyping approach in evolutionary development as a movement away from viewing systems development as a purely technical process. It is conceived as mostly a social process, grounded in an explicit theoretical framework which is sensitive to the organisational and broader social

context of information systems development. The planned organisational change theory takes a process-oriented view to the evolutionary development by concentrating on "means" to achieve systems objectives. Issues like institutionalizing the change process, conceptual training of users, role clarification and responsibility negotiation are the main interests of the theory in managing an evolutionary process. Semiotic theory by employing semantic analysis technique aims at formalising "ends" in information systems development through creating a common understanding among users. The findings of an explanatory case study, reported in chapter 6, demonstrated that the technique supports the above-mentioned issues and suggested, with certain qualifications, that the two theories can complement each other in providing a theoretical framework for evolutionary information systems development. Although the ultimate validation of the research is not possible to achieve, demonstrating the harmony between two theories might be taken as an indication of an attempt towards a subjective justification of the research within its context and assumptions.

On the basis of the proposed perspective, a method as an improved and well-defined description of a new approach to evolutionary development process is also developed. The method is documentable through ontology charts. The proposed method covers systems analysis as the process of collecting, organising, and analyzing user requirements and system design as the process of conception, generation and formation of a new information system. So, it mainly focuses on the middle stages of the life cycle model in information systems development. The main features of the method which seems to respond to the main difficulties of the evolutionary prototyping approach outlined in chapter 2 are:

- **Clustering of requirements:** the semantic analysis technique provides the constructs necessary for categorising and clustering user requirements descriptions into different subject areas, so that complexity can be reduced in performing design tasks.

- **Selection of requirements for prototyping:** the conceptual modelling formalism of the semantic analysis suggests criteria for selecting and

rejecting choices for prototyping according to the level of ambiguity for each subject area under investigation. It offers the designer a control strategy in the creation of the object system.

• **Feedforward process and mutual understanding:** The planned change model suggests as much feedforward process as possible to reduce complexity and uncertainty. By focusing on language norms of the workplace, the semantic schema caters for a better mutual understanding to take place among users. The requirements, captured by the semantic schemata, can be implemented using the evolutionary prototyping during the feedforward process.

• **Feedback process and requirements traceability:** The software prototypes can be traced back to the original semantic schema. Therefore, developers are able to use the results of actions to overcome uncertainty in feedforward paths. This enables solutions to be obtained even when knowledge of the environment is incomplete.

• **Learning process:** The conceptual training feature of the unfreezing stage and feedback process of the refreezing stage in the proposed perspective facilitate the addition of knowledge. To deal with the uncertainty of the world, the partnerships of users and designers must be able to adapt. Design performance can increase as user and designer build mutual understanding about the problem.

The findings of the explanatory case study, conducted in a management consultancy company, have shed light on some of the characteristics of the method. An explanation of the strengths and weaknesses of the proposed method can be drawn from the results of the explanatory case study and the assumptions behind the support theoretical framework.

One major strength of the method is that objective for design and use of information

systems is set on eliciting the design purposes and modes of use and on helping all users to understand and accept them. The assumption behind the method is that various users are adhering to different perspectives. By centering on the conceptual training of all users, the method focuses specifically on vehicles and tools to facilitate sense-making. The method demands a change in the perception of the role of the system analyst from an expert in prototyping approach to a catalyst who smooths the transition between evolutionary stages for the information system of which he is a part.

The exploratory case study showed that prototyping approach tends to reify systems requirements by suppressing their human authorship. The proposed method suggests that systems requirements and constraints are socially constructed; they change as perceptions change and perceptions change through continuous social learning and the evolution of language. The proposed method rejects the idea of validation of prototype systems by representative participants; i.e. User Forum as explained in the exploratory case study reported earlier in chapter 4. It adapts the view of consensus participation which believes any goals or values for information systems development are legitimate as long as they are consistent with social acceptance of the users. Information systems is concerned with the creation and sharing of meanings to legitimate social actions.

One important weakness of the proposed method is the complexity and somewhat overwhelming and opaque vocabulary of the semantic agent-based formalism employed in the semantic analysis and its bulky documentation. Another major issue which needs further clarification and elaboration is what are the actual impacts of conceptual models (explanatory prototypes) developed in the proposed system on behaviours in the business environment. We need more empirical and practical guidelines as to which representations are relevant and result in changes in behaviours with various individuals, or groups in different situations. It is not wise to assume that once the semantic schema has been agreed upon all problems of different interpretations disappear. The explanatory case study showed that the method is capable of providing conditions for increasing mutual understandings, but the question remaining of how convergent these understandings are to each other and how stable. When time and resources foreclose, action is taken on the basis of current understanding but not dictated by some formal

specifications. Therefore, communication through an evolutionary process facilitated by an explanatory prototype system can result in a process of interpretation and reinterpretation. This, however, is crucial in determining the boundaries of semantic analysis and in assessing the potential value of the proposed method.

The research approach did take note of institutional barriers to the rationality of communication and specifically the issue of organisational power, but excluded it from this study. Relying on "user's work language", the proposed method presumes that all users want to communicate. But this assumption seems unrealistic. Improving communication rationality requires addressing both the improvement of mutual understanding and the improvement of the conditions which shape the general arena of communication. Although the method responds to the former issue, it requires more investigation and completion of its theoretical framework and supporting techniques to address the later issue.

Finally, we expect that the method proposed here to be of practical value in the development of evolutionary information systems, but with perhaps a greater focus on a contingent approach to its utility. We do not expect to see the emergence of any standard method, but we anticipate the adoption of our method, despite the "backlash" against information systems development methods. We think that the method proposed is a signpost for an alternative way of looking at support methods for design understanding in information systems development in future. However, predicting the future is always problematical and has a habit of making fools of those who attempt it!

7.2 Recommendations for future research

There are at least four avenues for future research to extend the research of this thesis. The first possibility is that the future researchers could choose to replicate the validation stage of the proposed method. Replication could serve to check the findings of the explanatory empirical study of this research across different organisations. The goal would be to determine whether the semantic schema could be created from a different set of organisational subjects. Such replication would require identification and

examination of a relatively large number of organisations. It is also possible to attempt to validate further the planned change model in order to enhance it to keep pace with organisations' changing information requirements. Action research might be appropriate in this case. It is obviously hoped that this type of research will follow, as the method is intended for application. That is to say, it is the intention behind this thesis to suggest a method for information requirements determination that can be used in practice, especially in medium to large businesses.

Another possibility for future research related to this thesis is to implement a computer support system for the semantic modelling. By doing so, when the proposed method is applied to build a subject catalogue of an enterprise information model, a software prototype of each subject area would be built as a logical by-product. In that manner, the semantic analysis technique and the method created by this thesis would fuse with the system prototyping approach. Thereby, the method would improve as a result.

The third avenue for research related to this thesis, and possibly in parallel to the second avenue mentioned above, is to create an expert system based on the method proposed in this research. The knowledge pertaining to how the semantic model is applied for an organisation through semantic analysis could possibly be formalised into a knowledge-base and a set of rules. An inference engine could then draw upon that knowledge-base and those rules to execute the method proposed here. The nature of such a symbiotic partnership of user and designer for performing system design can be guided by a knowledge-based system which accommodates the method proposed here.

We believe that common information requirements exist because of common factors which shape organisations. Therefore the fourth possibility for related research in the future is to explore the isomorphic and homologous aspects of information requirements that are common to organisations. Given this situation, it follows that those aspects can be represented in universal semantic models. It is possible that we might be able to create a library of universal subject catalogues which incorporate the common elements of enterprise information models in different companies. The robustness of the semantic schema might allow us to model some universal aspects of organisations and

collect them in a form of a library of subject catalogues. This organisational knowledge-based library could open the prosect of analysis and design reuse in information systems development.

References:

Agresti, W. W. (1986). New Paradigms for Software Development. Washington: IEEE Computer Society Press.

Ahituv, N., & Neumann, S. (1990). Principles of information systems for management (3rd ed.). Dubuque, Iowa: Wm. C. Brown Publishers.

Alavi, M. (1984). An assessment of the prototyping approach to information systems development. Communications of the ACM, 26(6), 556-563.

Albadvi, A., & Backhouse, J. (1994). Implementing Business-Wide Applications. In 4th Annual Business Information Technology Conference(BIT'94), (pp. 108-119). Manchester, UK: The Manchester Metropolitan University.

Albadvi, A. (1995a). Developing information systems in business change environment. The British Computer Society, 3rd Annual Conference on Methodologies, 3. North East Wales Institute, Wrexham, UK: The British Computer Society.

Albadvi, A. (1995b). Engineering semantic traceability in object-oriented design. In K. Akingbehin & S.Y. Shin (Ed.), 13th Annual International Conference of the Association of Management, 13(1) (pp. 1-14). Vancouver, British Columbia, Canada: The Association of Management.

Albadvi, A., & Lee, H. (1996). What time is it? : A semiological analysis. In C. Fidler (Ed.), 14th Annual International Conference of the Association of Management, 14. Toronto, Canada: The Association of Management.

Alter, S., & Ginzberg, M. (1978). Managing uncertainty in MIS implementation. Sloan Management Review, Fall, 23-31.

Alter, S. A. (1980). Decision support systems-current practice and continuing challenges. Reading, Mass.: Addison-Wesley.

Andersen, P. B. (1990). A theory of computer semiotics. Cambridge: Press Syndicate of the University of Cambridge.

Andersen, P. B. (1991a). A semiotic approach to construction and assessment of computer systems. In H. E. Nissen, H. K. Klein, & R. Hirschheim (Eds.), Information systems research: Contemporary approaches and emergent tradition, IFIP (pp. 465-527). North Holland: Elsevier Science Publishers B.V.

Andersen, P. B. (1991b). Computer Semiotics. Scandinavian Journal of Information Systems, 3, 3-30.

Andrews, C. (1983). Prototyping information systems. Journal of Systems Management, 34(9), 16-18.

Angell, I. O., & Smithson, S. (1991). Information Systems Management: Opportunities

and Risk. London: The Macmillan Press Ltd.

Appleton, D. (1983). Data-driven prototyping. Datamation, November, 259-268.

Argyris, C., & Schon, D. A. (1986). Organisational Learning: A theory of action perspective. Reading, MA: Addison-Wesley.

Armour, F. J. (1993) A cluster analysis and prototyping approach for the risk management of software requirements. Ph.D. Thesis, George Mason University.

Avison, D. E., & Fitzgerald, G. (1995).Information Systems Development: Methodologies, Techniques and Tools. London: McGraw-Hill Book Company

Avison, D. E., & Wood-Harper, A. T. (1990). Multiview: An exploration in information systems development. Oxford: Blackwell Scientific Publications.

Backhouse, J. (1990) The use of Semantic Analysis in developing of Information Systems. Ph.D. (London)Thesis, Information Systems Department, London School of Economics and Political Science.

Backhouse, J., & Albadvi, A. (1994). Managing Business Change: Putting information systems in perspective. In H. Chang & L. Lu (Ed.), 12th Annual International Conference of the Association of Management, 12 (pp. 65-70). Dallas, USA: The Association of Management.

Backhouse, J., & Albadvi, A. (1995). A method for dynamic systems development. In G. Doukidis, B. Galliers, T. Jelassi, H. Kremar, & F. Land (Ed.), 3rd European Conference on Information Systems(ECIS '95), II (pp. 851-863). Athens/Greece: Print Xpress.

Balzer, R., T.E. Cheatham, J., & Green, C. (1983). Software technology in the 1990's: Using a new paradigm. IEEE Computer, November, 39-45.

Bartunek, J. M., & Moch, M. K. (1987). First-order, Second-order, and Third-order change and organisation development intervention: A cognitive approach. Journal of Applied Behavioral Science, 23, 483-500.

Bateson, G. (1972). Steps to an Ecology of Mind. New York: Ballantine Books.

Bjornestad, S. (1994). A research programme for object-orientation. European Journal of Information Systems, 3(1), 13-27.

Boar, B. H. (1984). Application Prototyping: A requirements definition strategy for the 80s. New York: John Wiley and Sons.

Bodker, S., & Gronbaek, K. (1991). Cooperative prototyping: users and designers in mutual activity. International Journal of Man-Machine Studies, 34, 453-478.

Boehm, B. W. (1976). Software Engineering. IEEE Transactions on Computers, December, 1226-41.

Boehm, B. W. (1981). Software Engineering Economics. Englewood Cliffs, New Jersey: Prentice Hall.

Boehm, B. W. (1984). Verifying and Validating software requirements and design specifications. IEEE Software, May(290-303).

Boland, R. J., Jr. (1978). The process and product of system design. Management science, 24(9), 887-898.

Booch, G. (1991). Object oriented design with applications. Redwood City, CA: Benjamin/Cummings.

Bostrom, R. P. (1989). Successful application of communication techniques to improve the Systems Development Process. Information and Management, 16, 279-295.

Bostrom, R. P., & Heinen, J. S. (1977). MIS Problems and Failures: A socio-technical perspective, part I: the causes. MIS quarterly, 1(3), 17-32.

Boynton, A. C., Victor, B., & Pine, B. J. (1993). New competitive strategies: Challenges to organisations and information technology. IBM Systems Journal, 32(1).

Brannen, J. (1992). Combining qualitative and quantitative approaches: an overview. In J. Brannen (Eds.), Mixing Methods: Qualitative and Quantitative Research (pp. 3-37). Brookfield: Avebury.

Brittan, J. N. G. (1980). Design for a changing environment. The Computer Journal, 23(1), 13-19.

Brooks, F. P. (1975). The Mythical Man-Month. Reading, Mass.: Addison-Wesley.

Brooks, F. P. (1987). "No Silver Bullet: Essence and accidents of software engineering". IEEE Computer, April, 10-19.

Burgess, R. G. (1982). Multiple strategies in field research. In R. G. Burgess (Eds.), Field research: A source book and field manual London: George Allen and Unwin.

Burgess, R. G. (1984). In the field: An introduction to field research. London: George Allen and Unwin.

Burn, J., Galliers, R., & Sauer, C. (1995). IT and the dynamics of organisational transformation(Panel discussion). In G. Doukidis, B. Galliers, T. Jelassi, H. Kremar, & F. Land (Ed.), 3rd European Conference on Information Systems(ECIS '95), II (pp. 1285-1288). Athens/Greece:

Carey, T. T., & Mason, R. E. A. (1983). Information systems prototyping: Techniques,

Tools, and Methodologies. INFOR - The Canadian Journal of Operational Research and Information Processing, August, 177-191.

Carpenter, D. A. (1992) Development of an information requirements determination methodology: Utilisation of normative analysis from a universal enterprise information model. Ph.D. Thesis, The University of Nebraska - Lincoln.

Chapman, P., & Talbot, S. (1996). ObjectStore: making Relational databases perform, Technical Report, Object Design UK.

Charette, R. N. (1986). Software Engineering Environment: Concepts and Technology. New York: McGraw-Hill.

Checkland, P. (1981). Systems Thinking, Systems Practice. Chichester: John Wiley & Sons.

Checkland, P., & Scholes, J. (1990). Soft systems methodology in action. Chichester: John Wiley & Sons.

Child, J. (1988). Organisation: A guide to practice and problems. Harper and Row.

Coad, P., & Yourdon, E. (1990). Object-Oriented Analysis (1st ed.). Englewood Cliffs, NJ: Prentice-Hall, Inc.

Coad, P., & Yourdon, E. (1991). Object-Oriented Analysis-OOA (2nd ed.). Englewood Cliffs, NJ: Prentice-Hall, Inc.

Colter, M. A. (1982). Evolution of the structured methodologies. New York: John Wiley & Sons.

Computer Weekly (1992). 7 May, 44-45.

Connell, J. L., & Shafer, L. I. (1995). Object-Oriented Rapid Prototyping. Englewood Cliffs, New Jersey: Prentice-Hall.

Cooper, R. B., & Swanson, E. B. (1979). Management information requirements assessments - The state of the art. DATABASE, 11(2), 5-16.

Couger, J. D. (1982). Evolution of system development techniques. New York: John Wiley & Sons.

Cox, B. J. (1990). Planning the Software Industrial Revolution. IEEE Software, 25(3).

Davis, A. M. (1990a). The analysis and specification of systems and software requirements. In R. H. Thayer & M. Dorfman (Eds.), System and software requirements engineering Los Alamitos, CA: IEEE Computer Society Press.

Davis, A. M. (1990b). Software Requirements: Analysis & Specification. Englewood

Cliffs, New Jersey: Prentice Hall.

Davis, G. B. (1982). Strategies for Information Requirements Determination. IBM Systems Journal, 21(1), 4-30.

Davis, G. B. (1974). Management information systems: Conceptual foundation, structure and development. New york: McGraw-hill.

Davis, G. B., & Olson, M. H. (1985). Management Information Systems: Conceptual foundations, structure and development. New York: McGraw-Hill.

De Brabander, B., & Thiers, G. (1984). Successful information system development in relation to situational factors which affect effective communication between MIS-users and EDP-specialists. Management science, 30(2), 137-155.

Dearnley, P. A., & Mayhew, P. J. (1983). In favour of system prototypes and their integration into the systems development cycle. The Computer Journal, 26(1), 36-42.

DeMarco, T. (1979). Structured analysis and System Specification. Englewood Cliffs, New Jersey: Prentice Hall.

Dennis, A. R., Burns, R. N., & Gallupe, R. B. (1987). Phased design: A mixed methodology for application systems development. DATABASE, 18(4), 31-37.

Denzin, N. (1970). The research act in sociology. London: Butterworth.

Desanctis, G., & Courtney, J. F. (1983). Toward friendly user MIS implementation. Communication of the ACM, 26(10), 732-738.

Dickson, G. W., & Simmons, J. K. (1970). The behavioral side of MIS. Business Horizons, August, 59-71.

Dickson, G. W., & Powers, R. F. (1973). MIS project management: Myths, Opinions and Reality. California Management Review, 15(3), 147-56.

Ding, C., & Mateti, P. (1990). A framework for the automated drawing of data structure diagrams. IEEE Transactions on Software Engineering, 16(5, May), 543-557.

Drucker, P. (1988). The coming of the new organisation. Harvard Business Review, Jan-Feb.

Fagan, M. E. (1976). Design and code inspections to reduce errors in program development. IBM Systems Journal, 15(3), 219-248.

Ferioli, C., & Migiarese, P. (1995). Opportunities and drawbacks of information technology in the emerging forms of organisation. In G. Doukidis, B. Galliers, T. Jelassi, H. Kremar, & F. Land (Ed.), 3rd European Conference on Information Systems(ECIS '95), I (pp. 559-573). Athens/Greece:

Fichman, R. G., & Kemerer, C. F. (1991). Object-oriented and Conventional analysis and design methodologies: Comparison and Critique (Working Paper No. CISR WP No.230. Sloan WP No. 3361). Center for Information Systems Research, Sloan School of Management, Massachusettes Institute of Technology.

Fields, N. A. (1991) An Evolutionary group decision model for computer supported cooperative work. Ph.D., George Mason University.

Financial Times (1993). 20 April.

Fitzgerald, G. (1991). Validating new information systems techniques: A retrospective analysis. In H.-E. Nissen, H. K. Klein, & R. Hirschheim (Ed.), IFIP TC8/WG 8.2 Working Conference on the Information Systems Research Arena of the 90's: Challenges, perceptions, and Alternative Approaches, . Copenhagen, Denmark: Elsevier science publishers B.V.

Floyd, C. (1984). A systematic look at prototyping. In R. Budde, K. Kuhlenkam, L. Mathiassen, & H. Zullighoven (Eds.), Approaches to Prototyping (pp. 1-18). Berlin: Springer-Verlag.

Floyd, C. (1987). Outline of a paradigm change in software engineering. In G. Bjerknes, P. Ehn, & M. Kyng (Eds.), Computers and Democracy - a Scandinavian challenge Aldershot: Avebury.

Fortune International (1993). 128(8), October 4.

Freeman, P. (1987). Software Perspective: The system is the message. Reading, Mass.: Addison-Wesley Publishing Company.

Galbraith, J. R. (1973). Designing Complex Organisation. Reading, Mass.: Addison-Wesley.

Galbraith, J. R. (1977). Organisation Design. Reading, Mass.: Addison-Wesley.

Galliers, R. D., & Land, F. F. (1987). Choosing appropriate information systems research methodologies. Communications of ACM, 30(11, November), 900-904.

Gibson, J. J. (1977). The theory of affordances. In R. E. Shaw & J. Bransford (Eds.), Perceiving, acting and knowing Hillsdake, N.J.: Lawrence Erlbaum Associates.

Giesecke, F. E. (1987). Engineering Graphics (4th ed.). New York: Mac-millan.

Ginzberg, M. J. (1978a). Redesign of managerial tasks: A requisite for successful design support systems. MIS Quarterly, 2(1), 39-52.

Ginzberg, M. J. (1978b). Steps toward more effective implementation of MS and MIS. Interfaces, 8(3), 57-63.

Ginzberg, M. J. (1981). Early diagnosis of MIS implementation failure: Promising results and unanswered questions. Management Science, 27(4), 459-478.

Gittins, D. T., Winder, R. L., & Bez, H. E. (1984). An icon-driven end-user interface to UNIX. International journal of Man-Machine Studies, 21, 451-61.

Goldstein, J. (1988). A far from equilibrium approach to resistance to change. Organisational Dynamics, 17, 16-26.

Golembiewski, R. T., Billingsley, K., & Yeager, S. (1976). Measuring change and persistence in human affairs: Types of change generated by OD designs. Journal of Applied Behavioral Science, 12, 133-157.

Goma, H., & Scott, D. B. H. (1981). Prototyping as a tool in specification of user requirements. In ACM/IEEE 5th International Conference on Software Engineering, San Diego:

Goma, H. (1983). The impact of Rapid Prototyping on specifying requirements. ACM Software Engineering Notes, April, 17-28.

Goodstein, L. D., & Bruke, W. W. (1991). Creating successful organisational change. Organisational Dynamics, 19(4), 5-17.

Greiner, L. E. (1972). Evolution and Revolution as organisations grow. Harvard Business Review, 50(4), 37-46.

Guimaraes, T. (1985). Study of application program development techniques. Communication of the ACM, 28(5), 500-5.

Habermas, J. (1972). Knowledge and human interest (Shapiro, J., Trans.). London: Heinemann.

Habermas, J. (1984). The theory of communication action. Volume I: Reason and the rationalization of society. Boston, MA: Beacon Press.

Hardcastle, A. (1994). Developing an executive framework to assess the suitability of information systems methodologies for specific information systems projects with a view to delivering business value. In 4th Annual Business Information Technology Conference (BIT'94), (pp. 197). Manchester, UK: The Manchester Metropolitan University.

Harel, D. (1987). A visual formalism for complex systems. Science of Computer Programming, 8, 231-274.

Hawgood, J. (1982). Evolutionary systems development. Amsterdam: North-Holland.

Hekmatpour, S., & Ince, D. C. (1986). Rapid Software prototyping. Oxford Survey in Information Technology, 3, 37-76.

Hirschheim, R., Klein, H. K., & Lyytinen, K. (1995). Information systems development and data modeling: Conceptual and philosophical foundations. Cambridge: Cambridge University Press.

Jeffrey, D. R. (1987). Software engineering productivity models for management information systems development. In Boland & Hirschheim (Eds.), Critical issues in information systems research New York: John Wiley.

Jorgensen, A. H. (1984). On the psychology of prototyping. In R. Budde, K. Kuhlenkamp, L. Mathiassen, & H. Zullighoven (Eds.), Approaches to prototyping (pp. 278-89). Berlin: Springer-Verlag.

Khan, B. K. (1985). Requirements specification techniques. Englewood Cliffs: Prentice Hall.

Kindler, H. S. (1979). Two planning strategies: Incremental change and Transformational change. Group and Organisational studies, 4, 476-484.

Kolb, D. A., & Frohman, A. L. (1970). An Organisational Development: Approach to consulting. Sloan Management Review, Fall, 51-65.

Korson, T., & McGregor, J. D. (1990). Understanding object-oriented: A unifying paradigm. Communication of the ACM, 33(9), 40-60.

Kraemer, K. L., & King, J. L. (1988). Computer-based systems for cooperative work and group decision making. ACM Computing Surveys, June, 115-146.

Kravshaar, I. M., & Shirland, L. E. (1985). A prototyping method for application development by end users and information specialists. MIS Quarterly, 9(3), 189-196.

Kristensen, B. B., Madsen, O. L., Moller-Pedersen, B., & Nygaard, K. (1991). Object-Oriented programming in the BETA programming language (Technical report No. Department of computer science, University of Aarhus.

Krovi, R. (1993). Identifying the causes of resistance to IS implementation. Information & Management, 25(North Holland), 327-335.

Lammers, S. (1986). Programmers at Work. MicroSoft Press.

Land, F. F. (1982). Adapting to Changing User Requirements. Information & Management, 5, 59-75.

Land, F. F. (1984). Critical assessment of software engineering. In D. Ince (Ed.), UNICOM Seminar on Software Engineering. Stevenage: Peter Peregrinus.

Land, F. F., & Somogyi, E. (1986). Software Engineering: The relationship between a formal system and its environment. Journal of Information Technology, 1(1), 14-21.

Lehman, M. M. (1981). The environment of program development and maintenance - programs, programming and programming support. New York: Computer Society Press.

Levy, A. (1986). Second Order Planned Change: Definition and conceptualization. Organisational Dynamics, 15(1), 5-20.

Lewin, K. (1952). Group decision and social change. In T. M. Newcomb, E. L. Hartley, & E. E. Maccobby (Eds.), Readings in Social Psychology New York: Holt, Rinehart & Winston.

Liebenau, J., & Backhouse, J. (1990). Understanding Information: An Introduction. London: Macmillan Education LTD.

Lucas, H. C. (1975). Why information systems fail. New York: Columbia University Press.

Lyytinen, K. (1987). Two views of information modeling. Information and Management, 12, 9-19.

Mahmood, M. A. (1987). Systems development methods - A comparative investigation. MIS Quarterly, 11(3), 293-312.

Malhotra, A., Thomas, J. C., Carroll, J. M., & Miller, L. A. (1980). Cognitive processes in design. International Journal of Man-Machine Studies, 12, 119-40.

Maude, T., & Willis, G. (1991). Rapid Prototyping, the management of software risk. London: Pitman Publishing.

McMenamin, S., & Palmer, J. (1984). Essential systems analysis. New York: Yourdon Press.

Meredith, J. (1993). Theory building through conceptual methods. International journal of operation & production management (IJO), 13(5), 3-11.

Moran, T. P. (1981). The command language grammar: a representation for the user interface of interactive computer systems. International Journal of Man-Machine Studies, 15, 3-50.

Mumford, E. (1983). Designing Participatively. Manchester: Manchester Business School.

Mumford, E. (1996). Systems design: Ethical tools for ethical change. London: McMillan Press Ltd.

Musa, J. D. (1983). Stimulating Software Engineering progress- A report of the software engineering planning group. ACM SIGSOFT-Software Engineering Notes, 8(2), 29-54.

Naumman, J. D., & Jenkins, A. M. (1982). Prototyping: The new paradigm for systems development. MIS Quarterly, 6(3), 29-44.

Nauta, D. J. (1972). The meaning of information. The Hague: Mouton.

Norman, D. A. (1983). Design rules based on analysis of human error. Communications of the ACM, 26(4), 254-8.

Nosek, J. T. (1984). Organisation design choices to facilitate evolutionary development of prototype information systems. In R. Budde, K. Kuhlenkamp, L. Mathiassen, & H. Zullighoven (Eds.), Approaches to prototyping (pp. 341-355). Berlin: Springer-Verlag.

Palmer, J. D., & Fields, N. A. (1992). An integrated environment for requirements engineering. IEEE software, May.

Palmer, J. D., & Aiken, P. (1990). Utilizing interactive multimedia to support knowledge-based development of software requirements, Center of Software Systems Engineering.

Palmer, J. D. (1988). Impact of requirements uncertainty of software productivity Centre of software systems engineering.

Parnas, D. L. (1979). Designing software for ease of extension and contraction. IEEE Transactions on Software Engineering, 5(2), 128-137.

Pine, B. J., Victor, B., & Boynton, A. C. (1993). Making mass customization work. Harvard Business Review, September-October, 108-19.

Pliskin, N., & Shoval, R. (1987). End-user prototyping: Sophisticated users supporting system development. DATABASE, 18(4), 7-17.

Pressman, R. S. (1987). Software Engineering: A practitioner's approach. New York: McGraw-Hill.

Riddle, W. E. (1984). Report on the software process workshop. ACM SIGSOFT-Software Engineering Notes, 9(2), 113-120.

Rieger, F., & Wong-Rieger, D. (1988). Model building in organisational/cross-cultural research: The need for multiple methods, indices, and cultures. International Studies of Management & Organisation (ISM), 18(3, Fall), 19-30.

Roberts, N., & Clarke, D. (1989). Organisational Information Concepts and Information Management. International Journal of Information Management, 9, 25-34.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., & Lorenson, W. (1991). Object-oriented modeling and design. London: Prentice-Hall International(UK) Limited.

Rzevski, G. (1984). Prototypes versus pilot systems: strategies for evolutionary

information system development. In R. Budde, K. Kuhlenkamp, L. Mathiassen, & H. Zullighoven (Eds.), Approaches to prototyping (pp. 356-367). Berlin: Springer-Verlag.

Sage, A. P., & Palmer, J. D. (1990). Software Systems Engineering. New York: John Wiley and Sons.

Saussure, F. d. (1966). Course in general linguistics. New york: McGraw-Hill.

Sayer, A. (1992). Method in Social science: A realist approach (2nd edition ed.). London: Routledge.

Scharer, L. (1983). The prototyping alternative. ITT Programming, 1(1), 34-43.

Schein, E. H. (1961). Management Development as a process of influence. Industrial management review, 2(2), 59-77.

Schein, E. H. (1972). Professional Education: Some new directions. New York: Mc Graw

Scott Morton, M. S. (1984). The state of the art of research. In F. W. Mcfarian (Eds.), The information systems research challenge Boston: Harvard Business School Press.

Searle, J. R. (1986). Minds, Brains and Science. Cambridge, Mass: Harvard University Press.

Smeltzer, L. R. (1993). Relevance is the issue. Journal of Business Communication, 30(4, October), 477-478.

Smith, M. F. (1991). Software Prototyping: Adoption, Practice and Management. London: McGraw-Hill.

Smith, T. J. (1993). READS: A requirements engineering tool. In IEEE International Conference on requirements engineering. San Diego.

Sroka, M., & Rader, M. H. (1986). Prototyping increases chances of systems acceptance. Data Management, March, 12-20.

Stamper, R. K. (1987). Semantic. In Critical issues in information systems research (pp. 43-78). Chichester: John Wiley & Sons.

Stamper, R. K., Backhouse, J., & Althaus, K. (1989). MEASUR- Method for Eliciting, Analyzing and Specifying User Requirements. In T. W. Olle, A. A. Verrijm-Stuart, & L. Bhabuta (Eds.), Computerized Assistance During the Information Systems Life Cycle North Holland.

Suchan, J. (1993). Response to Mohan Limaye: The need for contextually based research. Journal of Business Communication , 30(4, October), 473-476.

Szlichcinski, K. P. (1983). Designing for the day after tomorrow: I.The interaction between communications systems design and social change. Behavior and Information Technology, 2(3), 253-261.

Taggart, W. M., & Tharp, M. O. (1977). A survey of information requirements analysis techniques. ACM Computing Surveys, 9(4), 273-290.

Tavolato, P., & Vincena, K. (1984). A prototyping methodology and its tool. In R. Budde *et al.* (Eds.), Approaches to Prototyping (pp. 434-445). Berlin: Springer-Verlag.

Taylor, T., & Standish, T. A. (1982). Initial thoughts on Rapid Prototyping Techniques. ACM Software Engineering Notes, December, 160-166.

The Institute of Electrical and Electronic Engineers Inc. (1984). IEEE Guide to Software Requirements Specifications (ANSI/IEEE Standard No. #830). IEEE.

The IT management Programme (1994). Managing continuous change (Research report, Centre for management research.

Thonissen, K. (1990). Semantic Analysis: A study and comparison (Graduation Report, University of Twente, School of Management studies.

Turner, M. B. (1967). Philosophy and the science of behaviour. New York: Irvington.

Vogel, D. R., & Wetherbe, J. C. (1984). MIS research: A profile of leading journals and universities. DATABASE, 16(Fall), 3-14.

Wand, Y., & Weber, R. (1993). On the ontological expressiveness of information systems analysis and design grammars. Journal of Information Systems, 3, 217-237.

Watzlawick, P., Weakland, J. H., & Fisch, R. (1974). CHANGE:Principles of problem formulation and problem resolution. New York: W.W.Norton & Company Inc.

Welke, R. (1983). IS/DSS: DBMS Support for information systems development. In C. Holsapple & A. Whinston (Eds.), Data Base Management: Theory and Application (pp. 195-250). Reidel: Dordrecht.

Wetherbe, J. C. (1982). Systems Development: Heuristic or prototyping? Computerworld, 16(17), SR14-SR15.

Whitley, E. (1990) Embedding expert systems in semi-formal domains: examining the boundaries of the knowledge base. PhD, London School of Economics and Political science.

Williams, D. O. (1990) Developing systems for supporting design understanding. Ph.D. Thesis, University of Cambridge.

Williamson, O. E. (1979). Transaction-Cost Economics: The Governance of Contractual

Relations. Journal of Law and Economics.

Winblad, A. L. e. a. (1990). Object-oriented software. Reading, Massachausettes: Addison, Wesley.

Wolek, F. W. (1975). Implementation and the process of adopting managerial technology. Interfaces, 5(3), 38-46.

Woodhead, N. (1990). Hypertext and Hypermedia: Theory and Applications. Wuknskow, England: Sigma Press.

Young, T. (1984). Superior prototyping. Datamation, May, 152-158.

Yourdon, E. (1989). Modern Structured Analysis. Englewood Cliffs, New Jersey: Yourdon Press.

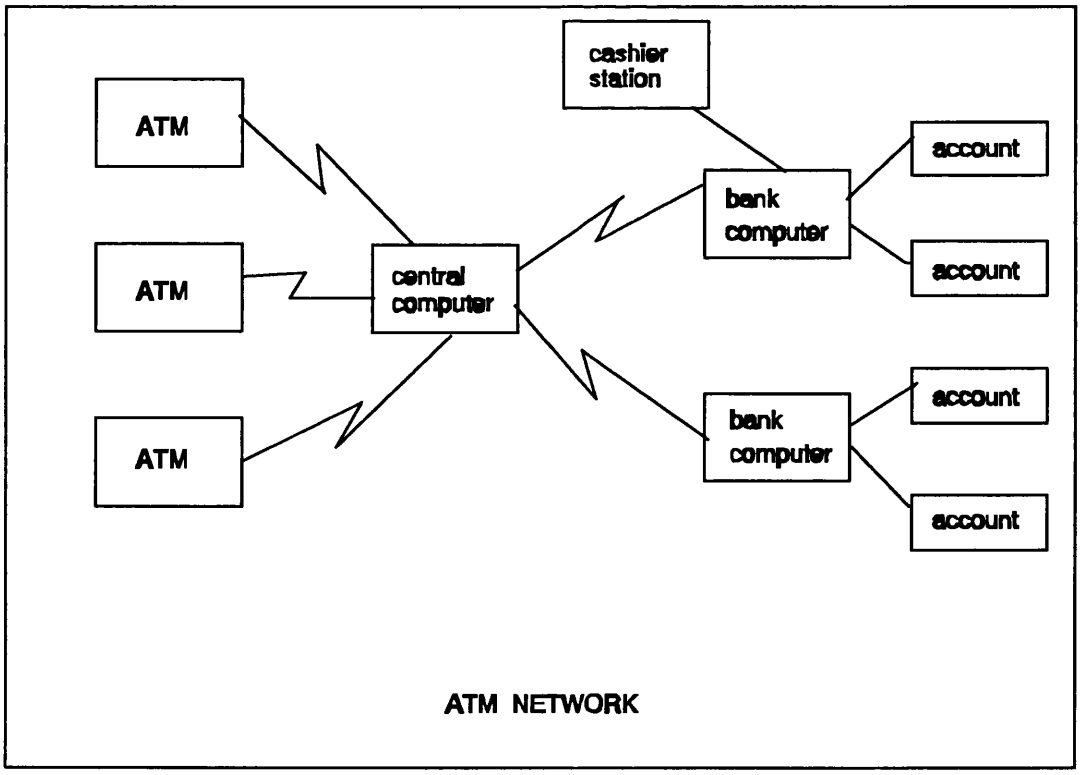
Zand, D. E., & Sorensen, R. E. (1975). Theory of change and the effective use of management science. Administrative science quarterly, 20(4), 532-545.

Zmud, R. W., & Cox, J. F. A. (1979). The Implementation process: A change approach. MIS quarterly, 3(2), 35-43.

Znaniecki, F. (1934). The method of sociology. New York: Farrars and Rinehart.

Appendix I. Automated teller machine example (adapted from: Object-oriented modelling and design, James Rumbaugh *et. al.*, p.151)

Problem statement: Design the software to support a computerized banking system including both human cashiers and automatic teller machines(ATMs) to be shared by a consortium of banks. Each bank provides its own computer to maintain its own accounts and process transactions against them. Cashier stations are owned by individual banks and communicate directly with their own bank's computers. Human cashiers enter account and transaction data. Automatic teller machines communicate with a central computer which clears transactions with the appropriate banks. An automatic teller machine accepts a cash card, interacts with the users, communicates with the central system to carry out the transaction, dispenses cash, and prints receipts. The system requires appropriate recordkeeping and security provisions. The system must handle concurrent accesses to the same account correctly. The banks will provide their own software for their own computers; you are to design the software for the ATMs and the network. The cost of the shared system will be apportioned to the banks according to the number of customers with cash cards.



Data Dictionary for ATM classes

Account: a single account in a bank against which transaction can be applied. Accounts may be various types, at least checking or savings. A customer can hold more than one account.

ATM: a station that allows customers to enter their own transactions using cash cards as identification, The ATM interacts with the customer to gather transaction information, sends the transaction information to the central computer for validation and processing, and dispenses cash to the user. We assume that an ATM need not operate independently of the network.

Bank: a financial institution that holds accounts for customers and that issues cash cards authorizing access to accounts over the ATM network.

Bank computer: the computer owned by a bank that interfaces with the ATM network and the bank's own cashier stations. A bank may actually have its own internal network of computers to process accounts, but we are only concerned with the one that talks to the network.

Cash card: a card assigned to a bank customer that authorize access of accounts using an ATM machine. Each card contains a bank code and a card number, most likely coded in accordance with national standards on credit cards and cash cards. The bank code uniquely identifies the bank within the consortium. The card does not necessarily access all of a customer's accounts. Each cash card is owned by a single customer, but multiple copies of it may exist, so the possibility of simultaneous use of the same card from different machines must be considered.

Cashier: an employee of a bank who is authorized to enter transactions into cashier stations and accept and dispense cash and checks to customers. Transaction, cash, and checks handled by each cashier must be logged and properly accounted for.

Cashier station: a station on which cashiers enter transactions for customers. Cashiers dispense and accept cash and checks; the station prints receipts. The cashier station communicates with the bank computer to validate and process the transactions.

Central computer: a computer operated by the consortium which dispatches transactions between the ATMs and the bank computers. The central computer validates bank codes but doesn't process transactions directly.

Consortium: an organisation of banks that commissions and operates the ATM network. The network only handles transactions for banks in the consortium.

Customer: the holder of one or more accounts in a bank. A customer can consist of one or more persons or corporations; the correspondence is not relevant to this problem. The same person holding an account at a different bank is considered a different customer.

Transaction: a single integral request for operations on the accounts of a single customer. We only specified that ATMs must dispense cash, but we should not preclude the possibility of printing checks or accepting cash or checks. We may also want to provide the flexibility to operate on accounts of different customers, although it is not required yet. The different operations must balance properly.

Verbs phrases:

Banking network includes cashiers and ATMs
Consortium shares ATMs
Bank provides bank computer
Bank computer maintains accounts
Bank computer processes transaction against account
Bank owns cashier station

Cashier station communicates with bank computer
Cashier enters transaction for account
ATMs communicate with central computer about transaction
Central computer clears transaction with bank
ATM accepts cash card
ATM interacts with users
ATM dispense cash
ATM prints receipts
System handles concurrent access
Banks provide software
Cost apportioned to banks

Implicit verb phrases:

Consortium consists of banks
Bank holds account
Consortium owns central computer
System provides recordkeeping
System provides security
Customers have cash cards