Expressing Business Rules: A Fact Based Approach

A thesis presented in partial fulfilment of the
requirements for the degree of

Master of Philosophy

in

Information Systems

at Massey University, Palmerston North, New Zealand

Adrian John Hargreaves

2004

Department of Information Systems

CERTIFICATE OF REGULATORY COMPLIANCE

This is to certify that the research carried out in the Masterate Thesis entitled "Expressing Business Rules: A Fact Based Approach" in the Department of Information Systems at Massey University, New Zealand:

(a) is the original work of the candidate, except as indicated by appropriate attribution in the text and/or in the acknowledgements;

(b) all the ethical requirements applicable to this study have been complied with as required by Massey University, other organisations and/or committees which had a particular association with this study, and relevant legislation.

Candidate's Name: Adrian Hargreaves

Signature: A Hargreaves

Date: 3/12/04

Supervisor's Name: Kevin Wilkinson

Signature: K Wilkinson

Date: 3/12/04

ABSTRACT

Numerous industry surveys have suggested that many IT projects still end in failure. Incomplete, ambiguous and inaccurate specifications are cited as a major causal factor. Traditional techniques for specifying data requirements often lack the expressiveness with which to model subtle but common features within organisations. As a consequence, categories of business rules that determine the structure and behaviour of organisations may not be captured until the latter stages of the systems development lifecycle.

A fact-based technique called Object Role Modelling (ORM) has been investigated as an alternative approach for specifying data requirements. The technique's ability to capture and represent a wide range of data requirements rigorously, but still in a form comprehensible to business people, could provide a powerful tool for analysts. In this report, ORM constructs have been synthesised with the concepts and definitions provided by the Business Rules Group (BRG), who have produced a detailed taxonomy of business rule categories. In doing so, business rules discovered in an organisation can be expressed in a form that is meaningful to both analysts and business people. Exploiting the expressive simplicity of a conceptual modelling technique to articulate an organisation's business rules could help to fill a significant requirements gap.

ACKNOWLEDGEMENTS

I would like to thank Tim and Gloria McGirr, proprietors of Mcgirr Training, for allowing me to conduct my research within their organisation. My thanks also extend to the staff of McGirr Training who provided me with valuable information and feedback.

The Office of the Deputy Vice Chancellor also deserves appreciation for awarding me the Advanced Degree Award. The award allowed me to devote a semester to my studies and resulted in the completion of four chapters of my thesis.

Finally, I would like to thank my supervisors Kevin Wilkinson and Claire Atkins for their informative feedback, advice and support during the last 3 years.

Table of Contents

List of Figures

List of Tables

List of Appendices

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

An essential role performed by the systems analyst is that of communicator. Perhaps the most important facet of this role is conveying the perceptions they have formed concerning a business system to the domain expert. In order to confirm the accuracy and completeness of the analyst's understanding of a system and its information requirements, the domain expert must be able to challenge those perceptions. Traditionally, analysts have relied on abstract models to capture the subtleties of business systems. Although these models are able to convey these details to other analysts, the domain expert is often less able to interpret the information they contain. But unless the content of these models is transparent to the domain expert, how are they to validate the perceptions of the analyst?

Many modelling tools and techniques also suffer from an inability to fully capture the data requirements of information systems. Although data structuring features such as sub-typing and generalisation can now be represented, the constraints that apply to these and other data structures are often weakly supported (ter Hofstede, Proper, & van der Weide, 1994). Where modelling approaches do consider such details, they are often expressed formally in the language of mathematics. Although formality adds rigour and precision to the data requirements captured, this approach is not likely to facilitate the involvement of domain experts in their validation.

In order to agree on what a business system currently does and what it actually should do, analysts require expressive modelling tools that capture requirements accurately and promote effective communication with domain experts. In the absence of such tools, one would expect such agreement to be difficult to reach.

1.2 SIGNIFICANCE

It is generally agreed that the analysis phase of the systems development life-cycle (SDLC) is of crucial importance to the overall success of IT projects. This is understandable, as a major deliverable of the analysis phase is a definition of the requirements for a business system. Unless errors and omissions within this definition are detected early, they often feed into successive phases of the SDLC. Unfortunately, incomplete, ambiguous and inconsistent requirements are commonplace in industry and these inadequacies often have a significant impact on software quality (Bell & Thayer T.A., 1976; Meyer, 1985). This suggests that approaches for capturing and representing requirements need to be improved in order to address issues relating to quality.

The scope of this thesis is restricted to the investigation of an approach for improving the transparency and expressiveness in which the data requirements of business systems are represented. To achieve this goal, a single framework involving the synthesis of a data modelling technique and a conceptual model of business rules will be developed. An expressive conceptual data modelling technique, known as Object Role Modelling (ORM), is used to represent categories of business rules as defined by

the Business Rules Group (BRG). The BRG have attempted to formalise an approach allowing business rules to be identified that define structural and behavioural properties of business organisations. Since ORM is able to verbalise assertions concerning business systems within a restrictive natural language, domain experts should be able to actively participate in the validation of business rules expressed in that language. By adopting this technique to articulate and define the data requirements of business systems, analysts may have an approach for improving the completeness, accuracy and quality of those requirements.

1.3 RESEARCH PROBLEM

The main focus of this thesis is to develop a conceptual framework for the articulation of business rules that define the data requirements of business systems. The aim is to provide an approach that allows analysts to work in close collaboration with domain experts in the definition of those requirements, thereby promoting an effective strategy for their validation.

Thus the problem to be resolved by this researcher is to determine whether ORM constructs can be used to articulate business rules in a form that domain experts can actively challenge.

The intention of this research is to address this problem in the following manner:

- Conduct a literature review that examines the problems relating to the definition of data requirements and approaches for resolving those problems.

- Synthesise ORM constructs with the business rules concepts and definitions formulated by the BRG, into a single conceptual framework for describing the structure and behaviour of business systems.
- Apply the synthesised conceptual framework within a New Zealand commercial organisation to define the data requirements for a new business system.

1.4 RESEARCH PROCESS

The steps of the above research process and the chapters of this thesis that relate to these steps are documented below.

Step 1. An investigation into the problems relating the specification of system requirements and the approaches adopted to resolve these difficulties.

Chapter 2 – Literature Review.

Step 2. Investigate and select research methods and describe how they were applied within this thesis.

Chapter 3 – Research Methods Selection.

Chapter 4 – Research Design.

Step 3. Develop a conceptual framework for the articulation of business rules.

Chapter 5 – Articulating Business Rules.

Step 4. Applying the framework within a commercial organisation in order to assess its efficacy.

Chapter 6 – Analysis of Findings

Step 5. Analyse the findings drawn from the application of the framework.

Chapter 6 – Analysis of Findings

Chapter 7 - Conclusions

1.5 THESIS STRUCTURE

The structure and relationship between the chapters within this thesis are described below.

Chapter 1: Introduction

The first chapter describes the significance and background of the research conducted, together with a discussion on the research problem and how it was investigated.

Chapter 2: Literature Review

The review of literature investigates previous research on the problems relating to the specification of system requirements and the approaches that have been developed in an attempt to resolve these difficulties. The chapter introduces conceptual modelling approaches, including ORM and the BRG's business rules model, and suggests that these approaches may be synthesised into a single framework to express the data requirements of business systems.

Chapter 3: Research Method Selection

Based on the conclusions of the literature review, the chapter investigates methods of research available to researchers with a view of selecting appropriate approaches for undertaking this thesis.

Chapter 4: Research Design

A detail account is provided on how the selected research methods were applied within a commercial environment to demonstrate the efficacy of expressing business rules using ORM constructs.

Chapter 5: Articulating Business Rules

The theoretical issues relating to this thesis are explored in this chapter. It is demonstrates that ORM has the ability to express all categories of business rules as defined by the BRG.

Chapter 6: Analysis of Findings

Having developed a single conceptual framework for the expression of business rules in chapter 5, its validity and efficacy are explored by applying the framework to define the data requirements of a new sub-system within a commercial organisation. The experiences of the researcher and domain experts in the application of this framework and the problems encountered are discussed in detail.

Chapter 7: Conclusions

A summary of the findings are presented and their relevance to the research problem stated in chapter 1 is discussed. Future research suggested by the undertaking of this study is also described.

Bibliography

Within this section, the references used throughout this thesis have been listed.

Appendices

The documentation and data models produced during the two case studies and action research component of the thesis have been included within the appendices, together the BRG's business rule model.

CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

Although computer systems have been developed for business and government organisations for more than 40 years, recent industry surveys (The Standish Group, 1995; OASIG, 1996; Whittaker, 1999; National Audit Office, 2001; Conference Board Survey, 2001) have all indicated that most I.T. projects still end in failure. Many of these surveys and the findings of other authors (Glass, 1997; Leffingwell, 1997) have concluded that inadequately specified requirements are one the primary causes of I.T. failures.

Perhaps these findings are not so surprising since many contemporary data modelling techniques simply cannot represent many important system requirements. In particular, the constraints that apply to data structures cannot be easily represented, if at all, in most data modelling techniques (ter hofstede, Proper, & van der Weide, 1994). Consequently, these requirements are often not discovered until the latter stages of the development lifecycle, possibly causing serious scheduling and budgetary problems for projects.

Not being able to represent the dynamic constraints that apply to static structural requirements could have other disadvantages. Capturing the business rules that define the structure and constrains the behaviour of business organisations has been

presented by some as a new paradigm for systems development. Indeed, Date (2000, p.3) states, "business rules can be seen in some respects as the next (and giant) evolutionary step in implementing that [original relational] vision."

The Business Rules Group (formerly known as GUIDE) has defined a conceptual model for these rules. The concepts and definitions within this report can guide analysts to seek appropriate information from domain experts by knowing "what is, and is not, a business rule" (GUIDE, 1995, p.1). Business rules originate as the policies adopted by organisations in response to constraints that act upon them (GUIDE, 1995). As such, business people represent domain experts and so it is from these people that analysts must elicit the organisation's business rules. Since these rules can only be validated when business people can challenge the analyst's understanding of them, it suggests that they must be expressed in a form that facilitates such dialogue.

Conceptual modelling approaches such as ORM and KISS, exploit the expressiveness of natural language within the formality of an underlying mathematical framework. Both of these modelling techniques rely heavily on natural language analysis. This allows the application model to be defined at the conceptual level where the analyst and domain expert can communicate more effectively (Halpin, 2002). Their ability to verbalise elementary facts that describe an organisation, suggests business rules could be expressed within these models. This would allow domain experts to become actively involved in their validation. By integrating conceptual modelling techniques within the business rules model, analysts may have a mechanism for capturing data

requirements more completely and accurately. That ability could help to fill a requirements gap and thus reduce the risk of project failure.

2.2 REQUIREMENTS ENGINEERING (RE) PROBLEMS

The problems associated with RE have been well documented for many years. Researchers have noted (Bell & Thayer T.A., 1976; Meyer, 1985) incomplete, ambiguous and inconsistent requirements are commonplace in industry and recognised the significance of these inadequacies on software quality. More recently, various studies (The Standish Group, 1995; OASIG, 1996; National Audit Office, 2001; Conference Board Survey, 2001) have confirmed these findings and the magnitude of the problems caused by poorly defined requirements. The Standish Group (1995) discovered that less than 20% of the 8000 American IT projects they surveyed, delivered a system that was complete, on budget and within schedule. One third of I.T. projects they surveyed were never completed. The remaining projects were only partially completed and often well over schedule and budget. Managers attributed the cause of over 40% of project failures to problems relating to the gathering, specifying and validating of requirements. These findings suggest that improving the quality of requirements can significantly reduce the risk of project failure.

RE is a complex task requiring the analyst to bridge the gap between two specialist perspectives on a system (Ryan, 1993). The domain expert views the system at a high conceptual level and it is a business-orientated perspective. The analyst must

transform this view into a complete and precise specification that is necessarily computer-system orientated (Pohl, 1993). Each of these perspectives is associated with concepts and a language used to express them. This presents a significant problem since in order to agree on what the required system should do, domain experts and analysts need to effectively communicate (Pohl, 1993). Unless the domain expert can challenge the analyst's understanding of a system, one would expect such an agreement to be difficult to attain.

2.3 APPROACHES TO REQUIREMENTS ENGINEERING

Modelling of the existing system seems to be a key task of RE regardless of which particular methodology is adopted. Models form the basis of communication about a system and its refinement to meet the needs it has to fulfil. This suggests that the information contained in these models needs to accessible to domain experts so that a common understanding of the system can be shared and requirements agreed.

Pohl (1993. p.2) identifies two types of problems facing RE; "the original RE problems and the problems associated with approaches that attempt to resolve the original problems". In the next section, modelling techniques that have been used in RE are discussed. The focus of this discussion will be to examine what aspects they attempt to model and how they overcome the limitations of their predecessors.

2.3.1 Structured Approaches

One of the most popular and certainly one of the earliest techniques employed in RE are entity-relationships (ER) diagrams. This semi-formal technique attempts to model the data requirements of the system (Chen, 1976) and is often used in conjunction with structured analysis for defining processing requirements (DeMarco, 1978) and state transition diagrams for modelling user interaction (Wasserman, 1979). Although simple to use, each of these techniques only model a single aspect of the system and lack the expressiveness of other techniques that followed. ER diagrams in particular have a limited capacity to model the constraints on data. For example, exclusion, equality and subset constraints are typically weakly supported (Halpin, 2002), as indeed are many of the rules that apply to data structures (ter Hofstede, Proper, & van der Weide, 1994). ER diagrams include thinly disguised implementation concepts such as tables (entities) and foreign keys rather than concepts more familiar to and thus more verifiable by, domain experts. Lacking a linguistic foundation compounds this problem since concepts cannot be easily verbalised to domain experts.

2.3.2 Requirements Modelling Languages

The limitations of ER diagrams to represent real-world concepts as appropriate data structures, lead some researchers to investigate alternative modelling approaches. Classification, generalisation and aggregation are concepts usually associated with object-orientation, but these techniques were first introduced in RML (Greenspan, Mylopoulos, & Borgida, 1982). This requirements modelling language was also one of the first to include formal semantics allowing requirements to be expressed more precisely than is possible using semi-formal techniques.

2.3.3 Agent-Based Reasoning

The techniques described above focus on modelling the static, structural qualities of systems. The representation of more dynamic aspects of the system such as behaviour and the constraints determining that behaviour were developed through agent-based reasoning (Feather, 1987). Agents are assigned responsibilities for achieving goals and constraints that limit their behaviour. Interactions between agents take place via interfaces and the formal foundations of the technique allow reasoning about agent behaviour and responsibility. This formality is achieved using a requirements specification language known as ALBERT in which requirements are expressed in terms of formal statements in temporal first order formulas (Yu, Du Bois, Dubois, & Mylopoulos, 1995). The capture of dynamic system qualities allows a more complete picture of domain knowledge to be represented, but its explicit formal framework could cause problems if business people are expected to validate the resulting models.

2.3.4 Goal-Based Reasoning

Systems development is more than just automating the current system. Re-engineering of existing practices may also be necessary if they are not sufficient to achieve business goals. The concept of requirements completeness was first incorporated into models through the explicit representation of goals (Yue, 1987). Goal-based reasoning allows analysts to exam the mechanisms in place for achieving business objectives. More formal approaches have been suggested (Manna & Pnueli, 1992) based on temporal logic that enable the completeness and correctness of requirements in attaining goals to be proved mathematically.

2.3.5 Scenario-Based Approaches

Even domain experts can have difficulty in identifying organisational goals and may require help to develop a deeper understanding of the system before these details can be documented. Scenario-based elicitation can be useful in these circumstances or when more traditional modelling techniques have failed (Weidenhaupt, Pohl, Jarke, & Haumer, 1998). A scenario is defined as "a description denoting similar parts of possible behaviours limited to a subset of purposeful state components, actions and communications taking place among two or several agents" (Plihon, Ralyté, Benjamen, Maiden, Sutcliffe, Dubois et al., 1998, p.13). The analyst elicits these details as they walk through a particular process with the end user to understand its operations, agents and the event that triggers the process. Scenarios can be expressed formally using grammar-based modelling languages (Glinz, 1995). Semi-formal (Potts C., Takahashi, & Antòn, 1994) and informal approaches (Rolland & Ben Achour, 1998) can also be used based on structured and natural language notations, respectively.

2.3.6 Object-Orientated (OO) Approach

OO analysis combines a number of semi-formal techniques for capturing data (Class diagram, Object diagram) behaviour (Statechart diagram, Activity diagram) and interaction properties (Sequence and Collaboration diagram), which are now tending towards a standard set of notations (Rumbaugh, Jacobson, & Booch, 1999). The wide variety of techniques OO incorporates, allows it to be used from the modelling of requirements through to design activities. Improvements in the data structuring constructs available within Class diagrams over earlier data modelling techniques, allow more details to be represented including the constraints on data. However, these

rules must be added as textual annotations to the model rather than symbolising these details graphically.

Since these techniques represent an abstraction of OO programming concepts (Mylopoulos, Chung, & Yu, 1999), implementation details are often present within models. This suggests that such models can be difficult for domain experts to conceptualise because the mechanisms and concepts underpinning the technique are unfamiliar to them (Halpin & Bloesch, 1999).

Finally, unlike several of the other approaches discussed above, OO focuses on addressing *What* type questions and ignores important *Why* type concerns associated with the early stages of R.E. (Mumford, 1981).

2.3.7 Conclusions Regarding RE Approaches

A number of conclusions can be drawn from the examination above. There has been a trend towards more expressive modelling techniques that can represent subtle data structuring features such as sub-typing and generalisation. Other, more dynamic facets of the system such as behaviour and constraints can now be modelled in addition to simply static, structural details. However, where such details are considered, they tend to be expressed formally within a requirements language based on first order logic, such as ALBERT. While this adds rigour and precision, requirements expressed in this form could be difficult to articulate and validate with domain experts.

There has also been increasing recognition of the importance of domain knowledge such as organisational goals and the mechanisms used to achieve them. Approaches

have been developed to elicit that knowledge more effectively using scenarios and other informal techniques like natural language. However, it is equally recognised that formality is required in order to produce a complete, precise and unambiguous specification.

Requirements are represented in varying degrees of formality, depending on the approach adopted. The next section briefly identifies these and the raises the problem of how requirements are to be accurately translated between each of these representations.

2.4 THE REPRESENTATION AND TRANSLATION OF REQUIREMENTS

According to Pohl (1993, p.4) the aim of R.E. "is to transform an operational need into a complete system specification through an iterative process of definition and validation." The representation of requirements should reflect this transformation in a process that begins informally and ends as a formal specification (Pohl, 1993).

Informal specifications tend to involve natural language or simple graphics, such as rich pictures (Checkland, 1981), which are user-orientated and expressive, but often contradictory and imprecise. Despite their imprecision, informal techniques can provide a mechanism for exploring system properties that need be understood first before they can be represented more formally.

Semi-formal techniques include Structured Systems Analysis and OO. They are popular techniques and provide useful graphical representations of system requirements. Since they possess a mathematical foundation, requirements expressed semi-formally are more precise and less prone to ambiguity than requirements expressed informally. Despite this, requirements expressed using semi-formal approaches cannot be proved correct to the same degree as more formal approaches.

Formal specification languages e.g. Z (Potter, Sinclair, & Till, 1996) and VDM (Jones, 1990) and knowledge representation languages e.g. RML (Greenspan, Mylopoulos, & Borgida, 1986) provide a much more rigorous and precise mechanism for representing requirements but are more difficult to use than less formal approaches. The mathematical framework, based on first order logic that underpins formal languages does, however, allow deductive reasoning on requirements to prove their validity and completeness (Manna & Pnueli, 1992).

Unfortunately, there is little in literature that actually describes how analysts should translate requirements between these levels of formality. As Pohl, K. concedes that, although the knowledge expressed in different representation formats should be integrated to achieve consistency, "The relationship between formal and informal is much less understood" (1993, p. 10). Even techniques within the same methodology do not have a defined mechanism that facilitates the translation process. For example, how does the analyst transform a series of Use Cases into a Class Diagram? Certainly, no automated tool support is currently available to achieve this task.

Requirements specified using formal languages often have no informal precursor. However, since formal specifications simply prove that the program produced from such a specification will work correctly, this approach cannot guarantee that the program specified is in fact the required program (Le Charlier & Flener, 1998). Further, since the fundamental purpose of a specification is to transmit information, the specification "must be comprehensible in itself and therefore must be written in the only language adapted to this purpose: natural language." (Le Charlier& Flener, 1998, p. 281). But how can requirements specified in natural language be translated into more formal descriptions of what the system will do?

The next section of this review examines two similar but distinct approaches to RE that support this transition from natural language to a formal representation of requirements. In addition, their method of exploiting natural language demonstrates that requirements expressed in this form are not necessarily informal.

2.4.1 Conceptual Approaches to RE using Natural Language

Natural language (NL) is a valuable tool for communicating with domain experts (Nijssen & Halpin, 1989; Hoppenbrouwers, Hoppenbrouwers, & Vos, 1996; Mazza, Fairclough, & Melton, 1994). However, NL is currently not widely used in industry for two major reasons. Firstly, NL does not lend itself for the expression of implementation concepts such as data retrieval and storage (Kristen, 1994). Secondly, NL possesses an inherent tendency towards informality and imprecision (Gamut, 1991). Despite these apparent shortcomings, a number of NL approaches to RE have been developed as a mechanism for defining a conceptual model of a system. Two NL

approaches are discussed and how they attempt overcome the two problems highlighted above are explained.

2.4.2 The KISS Approach

The KISS approach begins with the collection of textual descriptions of the organisation. Grammatical analysis is then performed upon these with a view of extracting specifications that will be used in lower level modelling (Kristen, 1994). Throughout this process, NL is extensively employed as the means of communication with domain experts.

Originally, experienced analysts performed grammatical analysis manually. However, since heuristics were the only guide for analysts conducting this analysis, differences in approaches were bound to occur (Hoppenbrouwers, van der Vos, & Hoppenbrouwers, 1997). This reflects the problem of translation between informal and formal specifications previously identified. In addition, the textual descriptions elicited from domain experts often-contained details that were not pertinent in the context of conceptual modelling or were semantically inaccurate. These problems highlight the misconceptions and omissions concerning domain knowledge that even domain experts possess (Hoppenbrouwers et al., 1997).

These problems have been alleviated to some extent by the introduction of CASE tool support in the form of the Grammarlizer (Grammatical Analyser) (Hoppenbrouwers, van den Heuvel, Hoppenbrouwers, Weigand, & de Troyer, 1997). This tool supports semantic tagging; a process that facilitates the partial interpretation of text from which a set of elementary sentences are generated. From these elementary sentences, the

CASE tool constructs 'KISS-structured sentences' that only contain information relevant for the generation of KISS conceptual models. The analyst must still examine each KISS sentence to determine its relevance within a conceptual model of the domain, but once performed, the CASE tool will create a Subject Communication (SC) and Object Interaction (OI) Model.

The SC model focuses on *subjects*, human or machine, that exchange *messages* providing a useful mechanism for depicting data flow. A typical SC model sentence would be 'The bank forwards a statement to the customer.' The details captured in SC models are dynamic in nature and therefore need to change as the business evolves.

The OI model defines the action types that can be applied to a particular object type and the synchronisation of those actions. A typical OI sentence would be, 'The reservation of a seat from a booking agency.' In this example, the action 'to reserve' is applied to both the seat and the booking agency concurrently. OI models capture more static components of the domain and tend to remain stable even after significant changes in the organisation.

An important component of the Grammarlizer architecture is the lexicon that allows both lexical and semantic knowledge to be captured and used for paraphrasing in NL. Currently, the Grammarlizer uses WordNet; a generic lexicon developed by Princeton University (Fellbaum, 1998), as a repository of all domain terminology and related linguistic elements. The lexicon contains the concepts and their relationship to other concepts in the domain and is therefore much more than a simple thesaurus or list of terms. Using NL paraphrasing as a validation technique, which exploits the domain

knowledge captured within the lexicon, assists analysts to overcome two conflicting concerns; developing a formal model of requirements and communicating these requirements in the language of the domain expert (Dalianis, 1995).

2.4.3 The ORM Approach

Object Role Modelling (ORM) is a well-established approach for conceptual data modelling and is associated with a methodology known as Natural language Information Analysis Methodology (NIAM), (Nijssen & Halpin, 1989) and Formal Object Role Modelling Language FORML (Halpin & Orlowska, 1992). A central concept within ORM is the idea of objects (entities or values) playing roles (parts in relationships). Unlike ER and OO modelling, ORM makes no initial use of the attribute construct and this approach tends make conceptual models more stable (Halpin, 1996). Instead, what would have been represented as attributes in other modelling approaches, are expressed as relationship types in ORM. As new facts are discovered, these details are simply added to the model as an additional role that is played by that object. Using an ER modelling approach, new discoveries concerning the application domain can mean that, concepts originally represented as attributes, need to be re-modelled as entity types (Halpin, 1996).

The application domain is modelled at a high conceptual level using terms familiar to domain experts in the form of elementary facts, constraints and derivation rules. Implementation issues, both logical and physical are ignored, allowing analysts to focus on the problem domain and to communicate effectively with domain experts.

Elementary sentences are constructed from discussions with domain experts that describe the roles played by objects e.g. 'Customer confirms Order'. In this example, there are two objects; 'Customer' and 'Order' that are associated via the role 'confirms'.

Natural language, in the form of elementary sentences, is often too ambiguous for further manipulation and therefore needs to be transformed into a notation with a formal mathematical framework. FORML (Formal Object Role Modelling Language) (Halpin, 2001) is the mechanism used to express elementary sentences formally in a process called verbalisation. Despite the mathematical foundations of this language, sentences constructed within this framework are still recognisable to domain experts. Using the above example, the equivalent FORML statement would be; 'Customer with CustNr 123 confirms Order with OrderNr 456.' The only significant difference between these two forms in this example is the use of a reference scheme for each object. The reference scheme gives a value that represents a unique instance of an object and provides a useful mechanism for validation against an actual population of such objects.

In the above example, the elementary fact is binary. However, ORM will support the expression of any type of arity. This allows facts to be expressed in more naturally, e.g. 'Customer with CustNr 789 at Time '9am' rents a Car with RegNr 'AB1234', represents a ternary fact type since it involves 3 objects.

Although a conceptual data model of the domain can be represented entirely within FORML statements, the ORM approach includes a step that allows analysts to

symbolise elementary facts graphically. This is a useful mechanism since diagrammatical representation has the advantage of conveying domain knowledge much more succinctly than simple verbalisation. This conciseness is bought at the expense of clarity, however. A graphical ORM representation would probably not be the appropriate tool with which to conduct validation by domain experts. In fact, as the number of requirements increases, the clarity of these graphical models diminishes rapidly (Feldman & Miller, 1986). This problem is known as the Database Comprehension Problem (Carlson, Ji, & Arora, 1990) and affects all 'flat' data models in which objects are viewed at a single level of abstraction (Campbell & Halpin, 1994). Since ORM explicitly represents concepts that would be modelled as attributes in an ER diagram, however, this Database Comprehension problem affects ORM more than most other graphical modelling techniques. To reduce complexity, it has been suggested that ORM models should be constructed using a number of levels of abstraction (Campbell, Halpin, & Proper, 1996). At the highest level of abstraction, a model would contain only the most important application details. These details are then successively decomposed into lower level diagrams in much the same way as data flow diagrams are levelled when using a structured approach.

ORM allows analysts to capture many more constraints, enforcing data integrity, than is possible is most other modelling approaches (Halpin, 2002). This ability allows analysts to explicitly represent these details at a conceptual level, not only facilitating validation by domain experts, but also allowing automatic code generation of these rules.

Uniqueness and role constraints have a parallel in all other data modelling approaches, but ORM allows many others to be expressed, such as:

Subset – where the population of one role is a subset of another

Equality – equivalent to a subset constraint, but in both directions

Exclusion – where the populations of two roles are mutually exclusive

Ring – applied to roles played by the same object i.e. recursive relationships. Includes asymmetric, intransitivity and acyclicity

Subtype – ORM also supports multiple inheritance

These constraints are added to the model as graphical annotations to structural details previously captured. As such, ORM appears to support the expression of many of the constraints that affect the behaviour of organisations. Details that are typically weakly supported, if at all, by many other modelling techniques (Halpin, 2002).

Details captured within ORM are mapped into table structures by using an algorithm incorporated into Visio Modeller, ORM's automated tool support. The algorithm transforms the conceptual model into Optimal Normal Form, which is equivalent to fifth normal form (Halpin, 2001). After mapping, the database can be generated into a number of target database formats including Oracle, Ingress and Access. Data integrity details, as specified by the constraints annotating the conceptual model, are also implemented in the target database. However, the degree to which these details appear as features within the target database are dictated by the limitations associated with the DBMS employed.

2.4.4 Conclusions Regarding Natural Language Approaches

Both KISS and ORM rely heavily on natural language paraphrasing techniques to communicate domain knowledge to domain experts. This is a useful approach in that knowledge is expressed in terms more familiar to those experts. The analyst's perceptions can then be actively challenged and thus validated by these people. Each approach resolves one of the major difficulties usually associated with the use of natural language: the inherent tendency towards informality and imprecision (Gamut, 1991). In both cases, this is achieved by adopting a restrictive grammar whose syntax and structure have a formal mathematical foundation. The KISS approach differs however, in the use of an automated tool (the Grammarlizer) that attempts to partially interpret textual information and transform these informal descriptions into formal expressions. By contrast, the ORM approach allows analysts to create graphical models, provide sample populations and facts, or to enter FORML statements directly into Visio Modeller. In this sense, precision and formality is achieved immediately, although still in a form comprehensible to domain experts, rather than being preceded by a phase where requirements are expressed informally.

The second problem associated with the use of natural language concerns its weakness for expressing implementation concepts (Kristen, 1994). In both the ORM and KISS approach, the domain is modelled at a high conceptual level in a fact-finding process. Implementation concepts, both logical and physical are simply not present within the models associated with each approach. However, Visio Modeller is still capable of generating databases from ORM models. Implementation issues are addressed internally, out of sight of even the analyst, by using algorithms to normalise ORM models and to implement them as relational tables and constraints (where

supported by the target DBMS). The KISS approach lacks the tool support to deal with these implementation issues and is primarily a technique for representing domain knowledge within a series of conceptual models.

Of the two approaches, only KISS attempts to capture semantic and linguistic knowledge using WordNet. WordNet can provide a mechanism that allows analysts to seek agreement amongst domain experts as to the particular meaning of domain specific terminology. Since these terms appear as the names of objects or roles within ORM models, it seems reasonable that capturing their meaning, as in the KISS approach, would add to completeness and consistency of these models (Hoppenbrouwers et al., 1997).

Neither approach addresses *why* type concerns by explicitly relating requirements to higher-level organisational goals. Both techniques could benefit by adopting mechanisms that address this issue, thereby improving the completeness of requirements they capture. Using goal and scenario-based approaches to augment these techniques could provide such a mechanism.

2.5 MODELLING BUSINESS RULES

Conceptual modelling techniques such as ORM and KISS distinguish themselves from many other modelling approaches in the degree of separation from technological aspects in which the application domain is described. Most other popular techniques, such ER and OO modelling define data requirements that incorporate implementation

concepts (Mylopoulos, Chung, & Nixon, 1992) not familiar to domain experts. By modelling the domain at a high conceptual level and using natural language to describe that domain, both KISS and ORM approaches allows people within these organisations to comprehend and challenge the analyst's understanding.

The business rules approach for defining organisations and their behaviour is possibly complementary to these conceptual modelling techniques. This approach describes the organisation in terms of 'the rules that define the structure and control the operation of an enterprise' (GUIDE, 1995, p.1). Since many of these business rules originate as policies implemented by organisations to achieve higher-level goals, the approach views the organisation from a similar conceptual perspective, independent of any technological considerations.

A project initiated by the Business Rules Group (BRG), consisting of E.F. Codd, Charles Bachman and John Zachman amongst others, had as its major objective the goal of "formalising an approach for identifying and articulating business rules", (GUIDE, 1995, p.1). They see these rules falling into one of four major categories; definitions of business terms, facts relating terms to one another (which they call structural assertions), constraints that limit the behaviour of organisations (referred to as action assertions) and derivations (calculations). As such, the business rules approach does not present any new concepts unfamiliar to system developers. However, by defining the organisation and its behaviour as a collection of atomic business rules, the approach provides a framework for describing the organisation that is meaningful to both analysts and domain experts.

The group achieved two main goals: to define and describe business rules and their related concepts and to define a conceptual model of business rules (see Appendix One). Within this framework it is now possible to 'express what is, and is not, a business rule' (GUIDE, 1995, p.1) guiding analysts to seek appropriate and complete information from domain experts.

Business rules themselves are declarative in nature, rather than procedural. They define states that are required or prohibited, which maybe conditional, rather than the steps taken to move from one state to another. As such, other, process-orientated techniques, are required to the define sequence in which they are executed.

The group argue that many techniques are available for representing the rules that define the structure of organisations (structural assertions), for example ER and OO approaches, but very few are able to describe how the behaviour of organisations are constrained (action assertions) (GUIDE, 1995). This argument is reflected in this review with only a few notable exceptions such as ORM, KISS and to a limited extent, UML. Consequently, action assertions are often neglected, but important requirements. These details may not be discovered until an attempt is made to implement these requirements as program code (GUIDE, 1995).

Recent industry surveys, such as that conducted by the Standish Group (1995), have identified inadequately specified requirements as the cause of 40% of all I.T. project failures. These surveys do not explicitly identify constraints as the source of these missing requirements. However, if analysts do not possess the tools to formally

articulate them, it seems reasonable to suggest that this inability could contribute to at least some of the problems encountered.

This raises an interesting possibility. Either ORM or KISS constructs could be used as the language to formally articulate the constraints (action assertions) on organisations, or indeed, any other types of business rule. Further, since these conceptual modelling techniques employ natural language, it suggests that these business rules can be expressed in terms familiar to domain experts. This would allow them to become actively involved in their validation. Thus, synthesising either ORM or KISS constructs with the BRG's business rule definitions into a single conceptual framework, may provide a mechanism for capturing requirements more completely and with greater accuracy. That ability could help to reduce the risk of project failure.

2.6 CONCLUSION

This review has highlighted that the IT industry faces a number of challenges that must be overcome if the current high level of project failure is to be addressed. Central to this problem is the inability of many popular data modelling techniques to represent important categories of system requirements and to express those requirements in a form meaningful to domain experts. It has been argued that unless business people can actively participate in the validation process, a common understanding of the system and its requirements would be difficult to achieve.

The business rules model provides a conceptual framework for understanding the structure and behaviour of systems within a paradigm meaningful to both analysts and domain experts. To apply the concepts contained within the model, this approach requires a language to accurately express business rules.

Conceptual modelling approaches such as ORM and KISS exploit the expressiveness of natural language within the formality of an underlying mathematical framework. Either of these conceptual approaches could be synthesised with the concepts and definitions within the BRG report. In doing so, they should provide a rigorous mechanism for expressing business rules. It is the focus of this thesis to make such an attempt with ORM to determine whether ORM constructs can be used for the expression of business rules.

CHAPTER 3

RESEARCH METHODS SELECTION

3.1 INTRODUCTION

The findings presented in the review of current literature, as detailed in Chapter 2, suggest that a significant cause of IT project failures are due to problems related to the quality of system requirements (Bell and Thayer T.A., 1976); (Meyer, 1985). The review discussed how traditional analysis techniques incorporate technical, implementation details unfamiliar to domain experts that may complicate an already complex task. The review suggested that because alternative, conceptual approaches to requirements engineering (RE) avoid implementation details, they may improve the essential communication between analysts and domain experts (Halpin, 2002). This led to the conclusion that quality of system requirements could also be improved if conceptual approaches to RE were employed that facilitated the ability of domain experts to challenge the analyst's perceptions of their system. The review identified two conceptual approaches that would be used to explore this conclusion; Business Rules Modelling (BRM) and Object Role Modelling (ORM).

In order to investigate the conclusions drawn in the literature review, a pluralistic method of research was adopted. This mixed method approach combines two qualitative techniques; action research conducted from within the framework of an explanatory case study. These research methods were complemented by grounded theory, to facilitate a rigorous, iterative analysis of collected data and triangulation, to confirm the validity of the data by drawing from multiple sources of evidence.

The case study draws upon qualitative data sources in the form of interviews with employees within a small Private Training Enterprise (PTE), observations of their business procedures and the documents used in carrying out those procedures. The aim of the case study was to discover the organisation's business rules[1] relating to the student mentoring process of their operation. The rationale for this new sub-system was to provide effective guidance to the organisation's students studying towards qualifications from the NZQA framework.

The literature review discussed the work of the Business Rules Group (BRG) who have defined a comprehensive taxonomy of business rules (GUIDE Business Rules Project, 1997) that guide analysts to seek appropriate information from domain experts. This taxonomy was adopted during this case study as a means of investigating whether this conceptual framework proved to be a useful vehicle for discussing the structure and behaviour of the PTE during interviews with employees.

Since business rules by necessity incorporate the terminology used within an organisation (GUIDE Business Rules Project, 1997), it is important that these terms are rigorously examined and their meaning standardised. Grounded theory, used in conjunction with triangulation, was applied to facilitate this process. Triangulation ensured that business terminology was defined by drawing from multiple sources of evidence from within the PTE organisation. These included interviews with different staff members, the documents and forms used by the organisation and glossaries of terms compiled by the New Zealand Qualifications Authority (NZQA). Grounded

[1] Refer to section 2.5 for a discussion on business rules and how they relate to this thesis.

theory provided the framework for systematically gathering and analysing these terms until a general consensus on their meaning within the PTE was reached.

Once the business rules and associated terminology employed by the PTE's student mentoring system were identified by the case study investigation, it was then necessary to transform these rules into a precise definition of the data requirements for that sub-system. Action research was used in a collaborative effort involving the target organisation's management, employees and the researcher to achieve this goal. The transformation from business rules to a specification of data requirements was achieved by expressing these rules in the formal natural language of Object Role Modelling (ORM). The review of literature identified this modelling technique as a conceptual approach for defining data requirements as a series of atomic facts (Halpin, 1993). Since the technique has the ability to verbalise these details in the form of natural language, domain experts are able to actively participate in the validation of the requirements they express. Since this process involved a transformation of the original business rules into a specification of data requirements, action research provided the collaborative framework in which these requirements could be validated by domain experts.

In order to assess the efficacy of expressing requirements conceptually in ORM's natural language, a second case study was conducted soon after the implementation of the student mentoring component. During this case study, management and end-users within the PTE were invited to make an assessment as to how closely this component matches their actual needs. If end-users were able to effectively participate in the validation of business rules articulated in this formal natural language during the

analysis phase, the quality of the data requirements these rules express should have been improved. If these improvements then result in the student mentoring system closely meeting the needs of the PTE, this suggests that conceptual approaches to RE deserve further investigation. The end-user assessment of this sub-system thus allowed the central research question of this thesis to be addressed; can ORM constructs be used as the language to express business rules in a form that facilitates effective end-user validation of these rules?

The selected research methods applied in this thesis are summarised in the following table:

Qualitative Research Method	Selection Rationale	How the Method was Applied	Data Sources Used
Explanatory Case Study (Interpretive)	The theory testing focus of the method.	1. To discover the PTE's business rules. 2. To test the hypothesis of this thesis.	Interviews, direct observation, business documents and procedure manuals.
Action Research	The collaborative focus of the method.	1. To transform business rules into system requirements. 2. Validation of these requirements by end-users.	Business rules discovered during the case study and the feedback from end-users.
Grounded Theory	Rigorous, iterative approach for analysing collected data.	To determine the shared meanings of the terms employed within the PTE.	Interviews with employees, NZQA glossaries, business documents and WordNet.
Triangulation	The ability to verify the repeatability of interpretations.	To ensure the validity of the meanings assigned to the PTE's business terminology.	Interviews with employees, NZQA glossaries and business documents.

Table 3.1 – Selected Research Methods

The remainder of this chapter describes some of the methods available to researchers, together with the rationale for selecting or rejecting those methods for inclusion within this thesis.

3.2 EPISTEMOLOGY

All research should possess an epistemological foundation since it is the assumptions about knowledge and how it is acquired that guides researchers as to validity of their inquiries and the appropriateness of their research methodologies. Orlikkowski & Baroudi (1991), based on the research of Chua (1986), discuss three perspectives through which research may be conducted; positivist, interpretive and critical.
The positivist perspective assumes that both the physical and social world exists independently of those who study it and can therefore, be objectively described and measured (Orlikkowski & Baroudi, 1991). Using this approach, the positivist researcher aims to empirically test theories and to discover causal relationships that can be generalised to larger populations.

Interpretive studies assume that people create subjective meanings as they interact with the world around them. Therefore, interpretive research adopts the premise that to achieve its goal of understanding phenomena one must first understand the meanings that people assign to them (Walsham, 1993). Understanding the context of phenomena is thus essential to understanding phenomena. In contrast with positivist research, generalisations from the research setting to larger populations are not a goal and so independent and dependent variables are not defined.

Finally, research can be conducted from a critical perspective, which assumes that people have unfulfilled potential and are able to modify the world around them (Orlikkowski & Baroudi, 1991). Cecez-Kecmanovic, (2001, p. 141) describes the critical perspective as seeking "to achieve emancipatory social change by going beyond the apparent to reveal hidden agendas, concealed inequalities and tacit manipulation involved in a complex relationship between IS and their social, political and organisational contexts". As such, critical research aims to critique the status quo by exposing the contradictions that exist in social systems.

These epistemological perspectives will be returned to later in this chapter where it will be argued that the interpretive stance is the most appropriate perspective from which to conduct the explanatory case study employed in this thesis.

It has been suggested that the significance of research is largely determined by the methods employed to conduct that research (Pinsonneault & Kreamer, 1993). Within the IS field, researchers have a wide variety of tools from which to draw (Cornford & Smithson, 1996); (Irani, Ezingeard, Grieve, & Race, 1999); (Walsham, 1995). At the broadest level, this includes both quantitative and qualitative approaches, which are discussed below.

3.3 QUANTITATIVE RESEARCH METHODS

Quantitative methods of research have dominated the study of natural phenomena since the 17th century. They are designed to be detached and independent of a specific

situation under study and as such, objectivity is of crucial importance when applying such methods. An important goal of quantitative research is that its findings can apply to more than one population; that is, the findings are generalisable. Gage (1994) describes the characteristics of quantitative research methods as follows:

> The ideals of quantitative research call for procedures that are public, that use precise definitions that use objectivity-seeking methods for data collection and analysis that are replicable so that findings can be confirmed or disconfirmed, and that are systematic and cumulative. (Gage, 1994, p. 372).

These characteristics of independence from the phenomena studied and the need for objectivity in the analysis of findings fits well with a positivistic perspective of knowledge that assumes that phenomena can be objectively described and measured. Consequently, quantitative research tends to adopt a positivist stance.

Although originating from the natural sciences, quantitative methods can be applied in any situation where the variables of interest are quantifiable, the formulation and testing of hypotheses is required and where it is reasonable to draw generalisations from samples to whole populations (Liebscher, 1998).

In the context of this thesis, quantitative methods could be applied by coding responses given within a questionnaire and then applying statistical techniques to analyse and draw conclusions from the data gathered. This approach could be used, for example, to determine the degree of end-user satisfaction with the student mentoring sub-system (referred to in section 3.1), whose requirements were developed during the action research component of the case study. This could provide a firmer foundation upon which to evaluate the effectiveness of ORM's language in specifying the PTE's business rules. However, since the number of employees in the

PTE is small, questions arise as to the significance these results would have in terms of generalising to larger populations. As previously stated, the ability to generalise research findings represents a significant goal and rationale for using quantitative research. To provide more certainty in this respect, the results from a number of similar cases would need to be analysed. However, since defining the data requirements for even a small system is a significant and time consuming activity, pragmatics dictate that this thesis should be restricted to a single case. Therefore, in view of the nature of the research undertaken and the consequences this has had in terms of limiting the number of cases, quantitative research was considered inappropriate for this thesis.

3.4 QUALITATIVE RESEARCH METHODS

Qualitative research methods have traditionally been used within the social sciences to facilitate the study of social and cultural phenomena. But with the developing interest in the organisational issues relating to computing technology (Benbasat, Goldstein, & Mead, 1987), IS researchers have found it increasingly useful to apply qualitative methods including ethnography, grounded theory, action research and case study.

3.4.1 Ethnography

Using this method, researchers seek to become a part of the social and cultural context in which the phenomena they are studying is set. The method demands that the researcher spends a considerable amount of time in the field in order to achieve this goal. From a pragmatic perspective, this could be seen as a disadvantage of the

method, particularly where constraints on time and access to the organisation exist. However, the approach has proved useful in IS research as a technique for incorporating multiple end-user and management perspectives into the design of a system (Holzblatt & Beyer, 1993).

An ethnographic study was considered as a research method for this thesis, since multiple perspectives in the appraisal of the work conducted in the target organisation will be sought. However, since the PTE is a small organisation with only about 20 employees, there were constraints on the time they could spend with the author to engage in such a study. Additionally, since the author is a full time employee in another organisation, work commitments dictated that an ethnographic study, although an appropriate method in many respects, would not be conducted.

3.4.2 Grounded Theory

In contrast with the generally positivistic methods associated with quantitative research, grounded theory does not aim to test hypotheses. Rather, theory should emerge, grounded in data that has been systematically gathered and analysed in an iterative manner (Strauss & Corbin J., 1994).

Sources of data used by this method include interviews, documents and observation. The method employs techniques and analytical procedures including the coding of collected data to identify and categorise central concepts and themes. This helps researchers to discover and remove inconsistencies within their data. The formalised nature of the method's techniques and procedures allows researchers to develop theory that is reproducible, generalisable and rigorous (Strauss & Corbin J., 1994).

Grounded theory has been used in I.S. research by several authors (Orlikowski, 1993); (Scott, 1998)) and more recently by Charmaz (2000), who advocated that grounded theory be applied in a more flexible, heuristic fashion.

Business rules by necessity incorporate the terminology of the organisation for which those rules apply. It is therefore important that standardised definitions of these terms were sought from employees within the PTE. Terms within this domain are defined with the aid of management, employees and a lexical database called WordNet[2]. It proved to be a useful tool for developing a shared meaning of the concepts used by the PTE.

Grounded theory and triangulation played their part in this process in the way in which the terms employed by the organisation were collected and analysed iteratively, until a consensus on their meaning was reached. By drawing on the perceptions from employees within the PTE in attempt to clarify the meaning of these terms, the technique of triangulation was incorporated into this analysis. Triangulation is defined as a process of verifying the repeatability of an observation or interpretation by employing multiple sources of evidence (Stake, 1995). WordNet acted as an automated tool support that facilitated this process. A discussion of this tool and how it was employed in the case study is described in section 4.3.9.

[2] This application is an on-going development by the Cognitive Science Laboratory at Princeton University (Fellbaum, 1998)

3.4.3 Case Study

Within the IS field, the increasing popularity of case study research has been discussed by several authors (Benbasat & Weber, 1996); (Klein & Myers, 1999); (Orlikkowski & Baroudi, 1991) and its efficacy has now been generally accepted (Klein & Myers, 1999). Case studies may be applied in numerous ways that often reflect both the underlying epistemology adopted by those who conduct them and the particular focus of their research. These themes are discussed in the next 3 sections.

3.4.3.1 Interpretive Case Studies

Several authors have provided definitions of case study research that reflect their epistemological perspectives. For example, Smith (1990) argues that a case study is "a way of organising social data so as to preserve its unitary character", thereby providing a richer, more complete understanding of research results. This reflects the interpretive stance supported by Walsham (1995), which recognises that IS concerns more than just technical issues; IS involves the application of technology within social systems (Kling, 1987).

Interpretive case studies generally draw upon several sources of qualitative evidence including interviews, observation (both participant and direct) together with artefacts and documents collected from the organisation being studied (Yin, 1994). The researcher must attempt to be as objective as possible when using these sources. However, it is recognised that subjectivity is impossible to entirely eliminate and therefore the researcher should be careful to identify their biases. This issue is addressed in section 4.2.5.4.

3.4.3.2 Positivistic Case Studies

Orlikkowski & Baroudi (1991) found that positivism is the prevailing epistemology in IS research and certainly, this is the perspective many authors apply to case studies. For example, Galliers categorises case study research as a "scientific approach being based on reductionism and refutability" (Galliers, 1992, p.154) and thus essentially positivistic in nature.

By applying purely positivistic methods within case studies based on numeric data collection and statistical analysis, as Galliers's definition would suggest, opportunities for rich contextual description may be lost. One approach that may exploit the merits of both the positivist and interpretive perspective would be to combine them within the same study (Kaplan & Duchon, 1988). In this way, researchers could apply interpretive techniques to explore contextual issues and their relationship to the phenomena under investigation and positivist methods to provide a firm foundation on which to base generalisations.

The positivistic perspective generally employs quantitative methods of data collection and analysis. Since the application of quantitative methods has been previously rejected as being inappropriate for this thesis (see section 3.3), a positivistic perspective for the case study was also rejected.

3.4.3.3 Critical Case Studies

Finally, case studies can be conducted from a critical perspective. The focus of such a study would be upon the oppositions, conflicts and contradictions present within the selected case and would aim to bring about emancipatory change. The main task then

of a critical case study is that of social critique. Since this is not the focus of this thesis, a critical case study was rejected as a possible perspective.

3.4.3.4 Selection of an Interpretive Case Study as a Research Method

An interpretive case study was chosen by the author as the means by which the PTE's business rules would be discovered. The rationale for selecting a case study was based on the need to acquire data to test the hypothesis of this thesis; that business rules can be expressed formally, but still meaningful to end-users, in ORM's natural language. A case study provides the opportunity to collect data within the natural setting of a business environment. Although it would not be appropriate to generalise the findings of a single case study to larger populations, the fact that these findings relate to a real organisation could provide the motivation for further research.

Adopting an interpretive perspective for the case study is thus based on the need to determine shared meanings and how language is applied to achieve common understanding of phenomena within organisations. This need arises as a consequence of the linguistic qualities of business rules, which use language to define the structure and behaviour of the organisation. These rules also incorporate the terminology of the organisation that is often specific to the environment in which they are applied. Since these business rules are to be transformed into ORM sentence structures that express these rules, which in turn are used to define data requirements, knowledge of how language is used in the organisation is crucial in defining requirements that are relevant. As the emphasis of interpretive research is to understand the meaning that people assign to phenomena (Walsham, 1993), it would appear that this perspective is the most appropriate for the case study employed in this thesis.

3.4.3.5 Case Study Categories

Not only can case studies adopt a positivist, interpretive or critical perspective, they can also be categorised as descriptive, exploratory or explanatory (Yin, 1993). Yin suggests that the particular category that a case study falls into is determined by the focus of that study. Descriptive case studies attempt to provide a complete description of an observed phenomenon within its context. For example, attempting to explain or analyse an organisation's I.S. strategy that resulted from a particular business action would categorise such a study as descriptive. Exploratory case studies attempt to define questions, constructs or hypotheses. Therefore, if in the previous example the investigation went a step further by trying to understand the vision and thinking behind the adopted I.S. strategy, such a study would categorised as being exploratory. Finally, the focus of explanatory case studies is on causal phenomenon facilitating the testing of a particular theory. The ability to conduct hypothesis testing suggests that this category of case study is of particular significance within IS research.

In order to determine which of the above case study categories represent the most appropriate form for this research, it is necessary to examine the focus of the investigation. The central question of this thesis is to determine whether the concepts and definitions associated with business rules can be synthesised with ORM, into a single conceptual framework meaningful to both analysts and domain experts. If this synthesis proves successful, this research will suggest that business rules can be formally expressed in ORM's natural language but yet still in a format that allows end-user validation of these rules. Sentence structures formed in this language can then be used as a basis for defining data requirements that are not easily captured, if at all, using traditional techniques. Further, since end-users can actively challenge the

analyst's perceptions of their systems, the quality of these requirements should be enhanced. These factors suggest that a theory testing focus to the case study is the most appropriate method to conduct this research and therefore an explanatory case study was undertaken.

3.4.4 Action Research

Action research is often described using Rapport's (1970, p.499) definition:

> Action research aims to contribute both to the practical concerns of people in an immediate problematic situation and to the goals of social science by joint collaboration within a mutually acceptable ethical framework. (Rapport, 1970, p.499)

In this context, the researcher is not merely an observer but a catalyst of change within the problem domain. Action research is also a learning process from which theory emerges in a collaborative effort between the researcher and the organisation in which it is conducted. A final consideration is that there should be a mutually satisfactory outcome between the researcher and the organisation.

Action research is less widely used in IS research than other methods, such as case study, because it is believed to suffer from a number of drawbacks. Baskerville & Wood-Harper (1996) have identified these as:

- The researcher's lack of impartiality,
- Its lack of discipline compared to other methods,
- It is often mistaken for consulting, and

- The difficulty of generalising findings

Despite these shortcomings, the same authors describe how action research may be effectively used within IS research (Baskerville & Wood-Harper, 1996).

Action research has been an established method of research in the social and medical sciences since the mid-twentieth century. A central argument of the method is that social processes are most effectively investigated by introducing change into those processes and then observing the outcome. According to Baskerville (1999), this contention of action research defines the method as interpretivist. This argument follows from the observation that the researcher intervenes and becomes a part of the study. The collaborative nature of action research results in the researcher's own values and *a priori* knowledge being incorporated into the study as they attempt to understand the meaning of their observations. In this way, the researcher adopts an interpretivist perspective. This viewpoint is also supported by Checkland (1981)who argues that the characteristics of scientific enquiry, such as reductionism, repeatability and refutation are not the ideals adopted by action research.

Lewin (1946) identifies two concerns that action research must address; supporting the production of practical outcomes and research theory. Achieving an appropriate balance between these two, possibly conflicting, concerns has lead to the suggestion that action research also raises control issues (Avison, Baskerville, & Myers, 2001). As a consequence of these characteristics, Mingers (2001) has argued that action research should be combined with other methods. In this way, the shortcomings of one method can be counter-balanced by the strengths of another.

The pluralistic approach to employing research methods could applicable in the context of this thesis. This applicability arises because of the nature of the investigation that is required to be undertaken in order to test the hypothesis of this thesis. In order to assess the efficacy of employing ORM constructs to express business rules, one must first discover what rules actually exist within an organisation. The case study briefly described in section 3.4.3.4 was used to facilitate the capture of business rules that exist within the PTE organisation. Having determined these rules, action research then addressed the need to determine the viability of formalising those business rules using ORM's sentence structures. The collaborative focus of action research, involving the employees of the target organisation and the author, was necessary to achieve this goal. The necessity for this collaboration is that business rules are typically informal in nature, whereas the facts expressed in ORM structures are formal. This implies that informal business rules must be transformed into formal ORM sentence structures. To help ensure that the resulting ORM sentence structures still accurately reflect the business rules they formally express, employees of the target organisation should assist in their validation.

The dual concerns of action research, generating change and generating knowledge, can be achieved in this study. The first concern of generating change will be addressed by helping to solve a practical problem by defining the organisation's data requirements for the student mentoring component of their system. The second concern, contributing towards theory, will be addressed by testing the hypothesis of this thesis; that business rules can be expressed rigorously, but still in a meaningful form for domain experts by using ORM sentence structures. For these reasons, action

research was considered to be a valuable research tool and was therefore incorporated into this thesis.

3.5 CONCLUSIONS

The underlying epistemology chosen for this research is that of an interpretivist perspective. Interpretive research adopts the premise that to understand phenomena one must first understand the meanings that people assign to them (Walsham, 1993). This thesis is concerned with the discovery of business rules within the target organisation and the meaning of the terminology incorporated within those business rules. Hence, determining a common understanding of these terms and rules is a significant goal of this research. Since interpretive research seeks to understand phenomena through social constructions such as language and because the use of language in the form of rules and terminology is central to this thesis, these factors suggest that an interpretive perspective could be successfully applied in this context.

Qualitative research methods in the form of action research conducted from within an explanatory case study have been employed. The case study uses primary, qualitative data sources in the form of interviews with employees within the target organisation, observation of their business procedures and the documents used within that organisation. These data sources help to define the business rules of the organisation that the case study examines. Grounded theory, supported by triangulation, assists in this process by providing a rigorous framework for the collection, analysis and validation of data. Action research, conducted from within the explanatory case study, promotes a collaborative setting to transform the organisations business rules into

ORM sentence structures. These structures can then formally define the requirements for the student mentoring sub-system of the PTE. This pluralistic approach of mixing research methods is thus used to determine the validity of this thesis's hypothesis as well as solving a practical problem within the target organisation.

CHAPTER 4

RESEARCH DESIGN

4.1 INTRODUCTION

In the previous chapter, the justification for using a combination of Case Study and Action Research was given and is assumed from this point onward. In this chapter, a detailed account of the research design and the manner in which research was conducted, are discussed.

4.2 RESEARCH DESIGN

This section has been divided into 5 areas that include:

- The purpose of the research,
- The research questions,
- The criteria used to select the case,
- The context of the case study,
- The attempts to limit the significance of bias.

4.2.1 Research Purpose

As stated in Chapter 3, action research, embedded within an explanatory case study, were chosen as the principle research methods with which to conduct this thesis. The explanatory nature of the case study facilitated the theory testing focus of the research, the purpose of which was to explore the hypothesis of this thesis that is

described in the next section. Initially, however, the focus of the case study was the capture of a Private Training Enterprise's (PTE) business rules for a new component of an existing system. The aim of this new component is to facilitate the student mentoring procedure that was previously difficult for staff to conduct effectively. Action research conducted within the case study was the vehicle by which those rules were formalised, thereby defining the data requirements for this new component of the current system. The collaborative nature of action research provided a useful framework to address a significant requirements engineering problem within the target organisation. Action research also contributed towards research by providing an environment in which conceptual approaches to requirements engineering could be explored. This was achieved by attempting to verbalise the target organisation's business rules in the formal, but natural language of Object Role Modelling (ORM). It is believed that expressing business rules in natural language will help to improve communication with end users, consequently leading to similar improvements in the quality of the resulting system requirements. A second, follow-up case study conducted after the implementation of the new sub-system component, attempted to access the efficacy of employing ORM constructs to specify business rules and thus test the central hypothesis of this thesis.

4.2.2 Research Question

The central question of this research is to determine whether Formal Object Role Modelling (FORML) sentence structures, an Object Role Modelling (ORM) construct, can be used to define sentences that accurately express the business rules of organisations in a form that facilitates effective end-user validation. In other words, can FORML be used as the language to express business rules that domain experts

can actively challenge? Once these business rules are formally expressed, they can be used as the definition of an organisation's data requirements for information systems.

The rationale for posing this research question is based on the fact that the majority of IS projects still end in failure, as detailed in section 2.2. Since a significant cause of failure is due to problems associated with the quality of system requirements, it suggests that mechanisms for improving their quality could reduce IS project failure rates. By employing conceptual approaches to capture and define data requirements, it is believed that their quality will be improved because domain experts are more able to challenge their validity.

4.2.3 Case Selection Criteria

A common criticism of case study research is the frequent use of a single case from which general conclusions are drawn (Yin, 1994). Since the unit of analysis in this research is a single case in one business organisation, this criticism needs to be addressed. It should be stressed that this research is not presented with a view of drawing generalisations that can be applied universally. It is recognised that a series of cases studies would need to be conducted, possibly incorporating quantitative analysis techniques, in order to confirm the findings of the case study presented within this thesis. However, this does not mean that a single case study is without merit. As Walsham (1995) suggests, some cases can provide a good foundation from which to make generalisations, especially in similar domains. But rather than claim that the case chosen falls into this category, the conclusions drawn from this research will be used instead as a justification for further study.

The rationale for selecting just a single case study as opposed to multiple cases is simply a matter of pragmatics. Determining the data requirements for even a small component of an existing system involves many hours of preparation, analysis, documentation and validation. The case study presented in this thesis involved over 200 hours of work within the target organisation and therefore it was considered impractical to conduct subsequent case studies to confirm or reject the findings presented. However, because of the reliance on a single case study, it is important that the case selected was carefully chosen. Although no attempt has been made to draw generalised conclusions from the case presented, it was still considered important that the selected case exhibited characteristics that can be identified in many other New Zealand businesses. To this ends, possible candidates for selection as a case were restricted to small to medium enterprises (SME) employing between 20 and 50 people and that had experienced significant problems with their information systems. In New Zealand, SME's employing between 1 to 20 people account for 97% of all businesses (Statistics New Zealand, 2003). Pragmatic considerations such as location of the organisation and the willingness and availability of management and staff to engage in the project were also important criteria.

The Private Training Enterprise (PTE) selected as the case for this thesis is a SME employing around 20 people. Also, like 88% of small businesses, the target organisation employs Information Technology in its day to day activities (Statistics New Zealand, 2002). The final justification for selecting this case was the difficulty the organisation has recently experienced with the systems it has developed. The PTE has expended over $80,000 NZD in the development of a student management system. However, after 18 months since the inception of the project, the system lacks

many of the features and functionality as was originally intended. Initial interviews with staff and the owner/managers suggested that the principal cause of these failures has been due to problems relating to requirements specification. This finding reflects a common problem with many IT projects, as detailed in section 2.2. Since the motivation behind this thesis is to determine the efficacy of a possible solution to such problems, the selected case presented itself as a good candidate to test the hypothesis of this research.

As a consequence of this organisation's characteristics and problems, it suggests that if improvements can be made to the quality of its data requirements resulting in an application that more closely matches the needs of the organisation, there is at least a possibility that similar findings may be found in other similar cases. This could provide the motivation for conducting further case studies in order to confirm or reject the conclusions of this thesis.

4.2.4 The Case Study Context

The target PTE selected for the case study is one of 50 such organisations that exist in the Wellington region, delivering programmes of study from the National Qualifications Framework (NQF). The NQF is a national qualification system administered by the New Zealand Qualification Authority (NZQA). PTE's compete with other such organisations to win contracts from the Tertiary Education Commission (TEC) who then provide the funding for a specific number of students. Having won a TEC contract, the PTE must then attract students onto its programmes.

The PTE comprises of about 20 employees, most of who are employed to deliver the organisation's programmes of study. In 2003, the PTE had a total of 140 students enrolled on its programmes. To support the activities of the trainers delivering the PTE's programmes, 6 administrative and technical staff are employed. Their duties include securing TEC contracts (most often performed by the two owner/mangers), financial and accounting functions carried out by the organisation's chartered accountant, secretarial and data entry functions and finally, two I.T. specialists who develop and maintain the organisation's computer systems. The organisation was established in 1996 and has grown steadily ever since. In 2002, they moved from the suburbs to the present location in central Wellington.

4.2.4.1 Significance of I.T. to the Organisation

Ever since the establishment of the organisation, I.T. has played a significant role both within its administrative functions as well as in the delivery of its programmes. The necessity for this latter role arises due to the organisation's specialisation in the delivery of business and computing qualifications from the NQF. As a consequence, the provision of networked computing facilities for the students is a necessary prerequisite for the delivery of such programmes.

The administrative role played by I.T. was initially one of automating the organisation's accounting functions, but with increasing student numbers, this role was extended to include student results processing and the generation of reports to meet their contractual obligations with TEC. Results processing and report generation software were initially written within MS Access, but in 2002, due to problems of scaling, a decision was made to rebuilt their information systems within SQL Server.

Visual Basic 5 was selected as the front-end of this DBMS as it was believed that this product would facilitate the development of more flexible and feature-rich user-interfaces.

It appears that the organisation's problems with its information systems began with this move to the MS SQL Server and Visual Basic environment. These problems can be summarised as increasing end-user and management dissatisfaction with the level of functionality displayed within the systems developed in this new environment. Possible causes of these problems include the increased complexity of their systems, in terms of hardware architecture and application functionality, and the procedures and techniques employed to determine requirements for these systems.

4.2.4.2 Timing and Duration of Case Study

The initial case study began in July, 2003. At this time, the migration to the new SQL Server/Visual Basic environment had been completed, but the problems outlined above were already being felt by end-users and management. This first case study was conducted over a 3 month period that involved spending 1 day a week within the target organisation and a further 10 hours per week working alone to prepare, analyse and document the study findings. Approximately 200 hours were spent conducting this case study, including the action research component, facilitating an in-depth analysis of the organisation's activities, problems and requirements for a new sub-system component. A second case study was conducted after the implementation of this new sub-system. Its purpose was to evaluate the effectiveness of employing the conceptual approaches advocated within this thesis and was conducted over a period of 3 days involving 8 hours of discussions with management and end-users.

4.2.4.3 Project Selection

During the initial meetings with the owner/managers, possible research projects were discussed. As a result of these meetings, two possibilities emerged. Either an existing but problematic sub-system component of the current system could be re-worked or the data requirements for a totally new component could be developed. The first option initially appeared the most appealing since it would provide opportunities for a direct comparison between the final products of each development strategy. If a sub-system component resulting from a conceptual development approach proved to be more effective in meeting the needs of end-users than the same sub-system developed using a traditional development methodology, this would be a strong argument in favour of adopting conceptual approaches in other, similar domains. Unfortunately, this option was eventually rejected as being impractical because no assurances could be given that existing database table structure would not require significant modification to accommodate the new design of the re-worked sub-system. Since all other sub-system components use this table structure, any significant changes to it would also result in modifications having to be made to all those components. The company's IT developer was particularly concerned that this possibility did not materialise.

As a consequence of this decision, it was also decided that the second option of developing a new sub-system component should be undertaken on the provision that it would not result in any significant modification to existing tables. However, the addition of new tables would be allowed since this would have little effect on the existing sub-systems. Although this initially appeared to be a significant constraint on the investigation, a possible project emerged that would satisfy the above constraint,

but still provide significant opportunities for research. The owner/managers were particularly interested in the development of a new component to their existing system that would allow more effective student mentoring. The aim of this sub-system would be to provide student supervisors with a mechanism that allows them to guide their students in terms of the options available to them to complete qualifications on the National Qualification Framework (NQF).

The NQF is a flexible but complex structure that allows students to 'mix and match' unit standards from the framework depending on the qualification requirements, the electives chosen by the student and their particular strengths and interests. A unit standard is a narrow field of study in which students attempt to demonstrate competency within one or more assignments. Competency is assessed according to pre-defined performance criteria that the student must meet within assignments to successfully complete the unit standard. In order for supervisors to provide effective mentoring, they need to know what unit standards the student has already achieved and what other combinations of unit standards are required to successfully complete the qualification as a whole.

To achieve this goal, the existing database table structure would require significant additions so as to allow the NQF to be mapped into their database. However, only a few changes to existing tables would be required, thereby satisfying the PTE's table modification constraint. The addition of appropriate new tables would enable the requirements of each qualification that the PTE delivers to be recorded into its database. Once completed, the specific requirements of a qualification could then be

matched against actual student achievement and the various pathways available for completion could then be determined and used to guide the student.

Although guidance is provided by supervisors in the above respect, there are currently no automated procedures to assist them in the current system. However, a basic facility does exist in the old MS Access-based system that preceded the current system, in a form of a report. The report provides a list of all the unit standards that the PTE currently offers. If a student has completed a unit standard from this list, the report indicates this fact as a tick against that particular unit standard. In the absence of any other facility, a number of supervisors use the old system for this purpose. However, there are significant problems with this approach, as listed below:

1. Student data has to be duplicated across these two systems.
2. The NQF is not mapped into the database; coded macros generate the lists.
3. Only the unit standards currently offered by the PTE are listed in reports.

Because of problems 2 and 3, this facility is inflexible to changing requirements. Qualification requirements and the content of unit standards are changed by the New Zealand Qualification Authority approximately once every 3 years. To reflect these changes, the code within the MS Access macros would need to be substantially modified. In addition, should the PTE offer new courses, additional macros would need to written so that any new unit standards offered by the PTE are listed in the reports.

All of the above problems could be avoided if the NQF structure were mapped as tables within a database. Student data would not need to be duplicated as there would be only one system, rather than two. Changes in qualification requirements or the

addition of new programmes of study would simple result in the modification or addition of new records within the database. As such, these changes could be made by the end-users, rather than specialist I.T. staff.

The PTE's rationale for not mapping the NQF into its database in either the old or new system is the complexity of the framework and the consequent difficult in determining the requirements to support such mapping.

The complexity of the proposed project and the perceived richness of the business rules that would be discovered in defining the data requirements necessary to implement this sub-system, led to this proposal being accepted as the basis for the case study.

4.2.5 The Attempts to Limit the Significance of Bias

The problem of bias in research is significant and ultimately impossible to entirely eliminate. Even within the physical and life sciences, bias manifests itself in the experiments chosen and manner in which they are conducted. Since some form of judgement is required in research, the question arises as to how these can be made without the researcher's own prejudices unduly influencing the process. Although techniques such as triangulation may be employed to limit the degree of personal bias, the researcher must also be explicit in what they are attempting to achieve and demonstrate how outcomes have been reached. The predispositions of the researcher and possibly the case study participants should also stated so that the reader is able to evaluate the degree of subjectivity, bias and lack of discipline. Further, by providing

the rationale for decisions the reader is able to evaluate the validity of an explanation given by the researcher.

Avison, Baskerville & Myers (2001) make a number of proposals that attempt to limit the degree of bias and subjectivity in research. These include:

- Drawing from multiple sources of evidence e.g. different informants, samples, case studies and researchers
- Iterative testing of interpretations of phenomena
- Seeking exceptions to any generalisations that have emerged and explaining them.

These proposals have been incorporated into the research design strategy as the following attempts to illustrate.

4.2.5.1 Multiple sources of evidence

The two major sources of potential bias that exist in the data collected from the PTE originate from the semi-structured interviews and the observation of business procedures conducted within the organisation. In both these situations, the potential for bias exists in both the supplier and recipient of data. The latter case and possible strategies for avoiding such bias were described in the previous section. Informant bias is discussed below.

It is generally accepted that the suppliers of data often attempt to provide details that they believe the recipient requires, rather than a more objective, fact-based

testimonial. This can be particularly relevant when there is a perception that disclosing certain information might reflect poorly on them or their superiors. In other circumstances, the inability of the subject to accurately remember events or the exact meaning of business terminology or the purpose of a specific step in a business process, could lead to 'imaginative' responses.

In an attempt to combat some of these problems, the technique of triangulation was incorporated into the collection of data. By seeking the same data from a number of different sources, it was possible to arrive at some degree of consensus. These sources included both different informants and, where possible, supporting documentation either from the PTE itself or from external agencies such as the New Zealand Qualifications Authority.

4.2.5.2 Iterative Testing

This was a particular relevant aspect of the research design when applied to documenting the business terminology discovered in the PTE, due the importance these terms have in the construction of business rules. Often, the terms used within organisations are specific to that domain and so the researcher is reliant on informants and other sources of data to provide an accurate and complete definition. To facilitate this process, grounded theory was applied so that definitions emerged from the data collected. Using the iterative approach that is the basis of grounded theory, each definition was either re-confirmed or modified accordingly after each cycle was completed. Multiple sources of evidence from different informants, internal and external documentation and the use of a lexical database called WordNet, provided the data for each successive cycle. These business terms and the evidence supporting

their definition were then presented to the management and end-users for final validation so that consensus on the shared meaning of each could be reached.

4.2.5.3 Exceptions to Generalisations

Contradictions were occasionally discovered during data collection and these were investigated in an attempt to determine their cause. These contradictions were often the result of misconceptions made by either the respondent or the researcher and were quickly resolved. In other situations, the contradiction revealed an exception to a general rule that then warranted further investigation so that it could be fully explained. These explanations were then challenged by using other sources of data to re-confirm or disconfirm the conclusions reached.

By seeking multiple sources of evidence, iteratively testing findings and investigating exceptions and contradictions, it is believed that the effects of bias in this research may be minimised.

4.2.5.4 Author Bias

Although the author has attempted to employ research tools in a manner that attempts to limit the influence of personal bias, it is accepted that such bias cannot be entirely eliminated. The author believes that the assertion of this thesis to be valid; that business rules can be expressed using ORM constructs, which in turn promotes end-user collaboration and improvements in the quality of data requirement. However, the purpose of this thesis is to prove this assertion by documenting the methods undertaken to gather evidence in support of this claim and to analyse the findings drawn from applying those methods.

4.3 THE CONDUCT OF THE CASE STUDY

In this section, the data collection methods and processes employed are discussed. This is recognised as a particularly important area of consideration for this research due to the reliance on a single case. This being so, there are possible implications for the reliability and validity of findings based on such a low unit of analysis. In an attempt to combat the limitations that a single case imposes, it was felt necessary to incorporate multiple data collection methods. This approach not only provides a richer picture of the events under study (Yin, 1994); (Sawyer, 2001), but also improves the confidence in the findings of this research. These data sources and the way they were used are discussed below.

4.3.1 Semi-structured Interviews

In common with many interpretive case studies, the investigation used face-to-face semi-structured interviews as the primary data collection method. As stressed by Kaplan and Maxwell (1994), the goal of interviews is to elicit the respondent's views and experiences from their own personal perspective, rather that simply collecting data that conforms to a choice between pre-defined responses. Although this approach presents challenges in terms of the analysis of data collected, semi-structured interviews do provide a forum for rich contextual description that could be lost using other methods.

4.3.2 Sampling Strategy

A key issue when using interviewing as a data collection method concerns the sampling strategy employed to determine who should be talked to and what should be

discussed. Interviewee selection is important since it places limits on the conclusions that may be confidently drawn from the data collected. Due the size of the organisation, it was not necessary to be overly concerned with strategies for obtaining representative samples. Since the PTE employs only 20 employees, it was certainly a possibility to interview everyone if this had proved necessary. Instead, a snowball sampling strategy (Strauss and Corbin, 1990) was employed. This strategy involves key members of the organisation being identified by the researcher and interviewed, who then in turn identify other people who should be interviewed, and so on. Using this approach, the two owner/managers and the two IT personnel were first interviewed. The responses given within this interview helped to set the direction of subsequent investigation and to determine who should be interviewed next. Thereafter, the selection criteria for potential interviewees was often dictated by the need to obtain an answer to a question that had been raised by the researcher, but could not immediately be answered by the current interviewee through lack of knowledge on that specific aspect of the organisation. To answer such questions, it was necessary to arrange another interview with a person who did possess the knowledge required to further the goals of the investigation. This process was continued throughout the case study until all of its goals had been achieved. This point was reach after 12 of the 20 employees in the PTE were interviewed.

4.3.3 Interview Planning

Before each interview was conducted, a schedule was drafted in which the goals of the interview were identified, together with a preliminary list of questions. Since the interviews were semi-structured, the direction of the interview often changed in unanticipated ways as new, unexpected information came to light. Although this

added richness and breadth to these discussions, it was occasionally necessary to re-direct the interviewee back to the general direction planned for that interview. A sample interview schedule has been provided below.

4.3.4 Example Interview Schedule

Goals:

To understand the student enrolment procedure and the extent to which the current system supports this process.

Questions:

1. What or who initiates the enrolment process?
2. What are the general steps involved in processing a students enrolment?
3. What exceptions can occur and under what circumstances?
4. What decisions need to be made during enrolment and how are they made?
5. What data is required to enrol a student and where is it obtained?
6. What data is recorded during enrolment and who subsequently uses that data?
7. How does the current IT system support the enrolment procedure?
    - Useful features provided
    - Problems (how are these worked around?)
    - Desired additional features

4.3.5 Validating and Documenting Interviews

Each interview typically lasted an hour and was initially tape recorded and later transcribed. However, a perception emerged soon after these interviews began that

many interviewees felt reluctant to discuss problems with the current system or to be critical about current business practices and procedures, when being recorded on tape. Consequently, field notes were taken in preference to recording these details for most of the conducted interviews.

If at all possible, more than one person was interviewed to provide a mechanism for ensuring the validity of the data collected. This involved conducting two or more interviews with the same goal and questions but with different employees. However, since the PTE is a small organisation, several roles within it are carried out by only one person. In these circumstances, other data sources were used to corroborate the findings within the interview. In all cases, the data recorded from the first interview was validated by the same interviewee in a second interview. This was considered necessary since on most occasions, the details recorded in the first interview were modelled graphically using a variety of process modelling tools or as verbalisations of business rules. The second interview provided a forum to validate these models and rules and another chance to capture details that may have been missed during the first interview.

4.3.6 Interview Focus

Semi-structured interviews were conducted in all stages of the investigation. Initially, the focus of the interviews was directed in placing the organisation in context with its environment. Constructing a context Data Flow Diagram (DFD) based on the findings of these initial interviews assisted in this process and proved useful vehicle with which to validate these findings with employees within the organisation. This exercise

also facilitated the process by which familiarisation with the organisation and its goals were initiated.

After conducting the initial interviews, it was decided that all of the organisation's business processes relating to students should be investigated. This was felt necessary because the proposed new student mentoring system would need to draw upon data collected and processed in most of the organisation's procedures involving students. Therefore, it was necessary to have a clear understanding of these procedures and the data they would supply to the new sub-system component. Also, since the PTE number fewer than 20 employees, it was considered a practical as well as a desirable goal. Interview schedules with management were planned to facilitate the construction of a Level 1 DFD so that each sub-system was identified together with details of its interfaces with other sub-systems. Again, using a DFD process model allowed the details of the interview to be summarised pictorially and was used with interviewees to validate the findings of these interviews. Although only the organisation's IT staff had been trained to interpret such DFD's, only a little guidance was required for other staff to appreciate the details they contained. However, it was not felt appropriate to use DFD's with end-users to validate processing details lower than Level 1, because of the perceived difficulties of clearly communicating these details with end-users. Instead, business process modelling, favoured by some business analysts, was used to document and validate lower level processing details discovered during the interviews. Business process modelling is a physical, rather than a conceptual modelling tool. However, the ability to document the roles of those carrying out process steps, the sequence in which those steps are performed, the decisions that were made and the triggering mechanisms for the process as a whole, enabled

employees to relate to these models and validate their content. Consequently, most employees required only a short training session to appreciate the significance of the information these models contained. In addition, since these models were constructed within Oracle Designer, models could be augmented with links to HTML or Microsoft Word documents that enabled the details of each process step to be expanded upon to aid clarification. This feature was also used to initially document business terminology as it emerged within the interviews and to record contextual information that semi-structured interviews tend to capture. Options to add colour and even to animate the models to highlight the sequence of steps performed in a process, are available in this automated tool. These process models and the DFD's that preceded them are documented in Appendix 2 and 1, respectively.

Having acquired enough data from these interviews to construct lower level process models and to validate them, these models were then analysed for their business rule content. At this stage, Visio Modeller was used first to document the business terminology that had been previously captured and later, the business rules themselves. Interviews at this stage of the investigation shifted in focus from data collection to data validation. The ability of Visio, ORM's automated tool support, to verbalise business rules made this a reasonably straight-forward task since the employees appeared to have little difficulty in challenging any misconceptions that arose. After transforming these rules into a detailed specification of data requirements during the action research component of the study, interviews again were used as the means for ensuring the validity of these requirements.

Once the new student mentoring component had been implemented according the specification of requirements, it was then necessary to assess the degree to which this sub-system meets the needs of the organisation. Determining this information gave some indication as to efficacy of employing conceptual data modelling techniques to specify requirements in a form that improves communication with end-users and thus the quality of those requirements. Semi-structured interviews with management and those employees who conduct student mentoring were selected for this evaluation. The semi-structured approach was again considered to be the most appropriate form of conducting the interviews, as it promotes opportunities to discuss central issues in depth and other issues that may not have been previously considered during the planning of the interview.

Although interviews represented the most important instrument for data collection in this investigation, other data sources were also used and are described below.

4.3.7 Direct Observation

This qualitative research method involves the researcher observing phenomena of interest within its context and is useful for providing additional information about the topic being studied. Often it is conducted during a field visit of a case study and could involve simple data collection activities or formal protocols to measure and record behaviour. An important consideration when applying this method is the need for the researcher to be as unobtrusive as possible (Glesne & Peshkin, 1992) so as to minimise the potential of unduly influencing the phenomena under study.

Direct observation was employed in this thesis to provide additional insights into the business procedures carried out by staff, supplementing the data collected during the semi-structured interviews. It also proved to be a useful way to validate the perceptions formed during those interviews and occasionally revealed new information not originally identified. This was particularly the case when the procedure under investigation involved the use of the existing IT system to carry out part or all of that procedure. In these circumstances, problems relating to the current system were occasional demonstrated by the end-user and sometimes they were able to demonstrate how such problems were worked-around. These problems were examined during the interviews, but direct observation often revealed the finer details of such problems that were not identified in the preceding interview. Direct observation was therefore a useful mechanism for helping to ensure the completeness of the investigation. It also provided another source of data allowing the technique of triangulation to be conducted and thus helped to ensure the validity of the perceptions drawn during the interviews.

4.3.8 Document Data Sources

These documents can include letters, memoranda, agendas, administrative documents, newspaper articles or indeed any other document considered relevant to the investigation (Yin, 1994). In the context of this case study, the documents used consisted of paper-based forms (both internal and external), the data-entry screens and reports associated with the current system, procedure manuals and on-line documentation from the NZQA website (New Zealand Qualifications Authority, 2004).

These documents were used in a variety of ways but mostly as a means of supplementing and validating the details captured within interviews. Procedure manuals, reports and data-entry forms were used to investigate the processes associated with student management and problems associated with the current system in carrying out these processes.

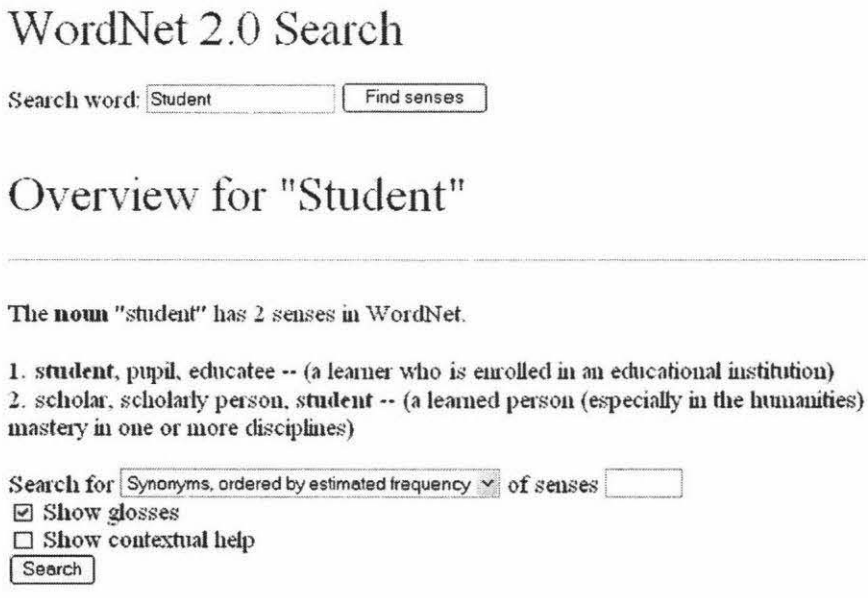
NZQA documentation was particularly useful in helping to define the data requirements for the new student mentoring component. Information contained on their website is structured to reflect the National Qualifications Framework (NQF) allowing a thorough investigation of not only the requirements associated with qualifications, but also where each qualification fits within this framework. Both of these details were crucial in determining the data requirements for the new student mentoring sub-system. The structural details of the NQF were required so that this framework can be mapped into the existing table structure of the current system. The structural details of the requirements associated with each qualification were also examined so that they too could be mapped. Reaching this goal allowed student achievements to be matched against the requirements of a particular NZQA qualification and provided a mechanism for supervisors to guide students towards the completion of the qualification.

Online documentation from the NZQA website also provided a comprehensive glossary of terms associated with the NQF. These details are important since the terms adopted within this domain are required to express business rules that described the structure and behaviour of the PTE. These terms and their meanings were compared against the terminology applied within the organisation in an effort to standardise

their usage. The technique of grounded research assisted in this process during which these terms were iteratively analysed so as to allow their meaning to emerge from the collected data. When a consensus had been reached with management and end-users, these terms were formally documented within Visio Modeller and later used to express the organisations business rules.

4.3.9 WordNet

More general terms not specific to the PTE or the NQF were defined with assistance from WordNet, an online lexical database developed by Princeton University. English nouns, verbs, adjectives and adverbs are organised into synonym sets, each representing one underlying lexical concept (Fellbaum, 1998). Using this tool, a word is simply entered into a text box and the senses for that word are returned from the lexical database as illustrated below.



This proved to be a simple, but effective tool for helping to define more generic terminology. However, the process of seeking a consensus from management and

end-users on the meaning of a term in the context of the organisation was still followed, rather than simply accepting the meaning provided by WordNet. This provided an opportunity to modify the standard definitions returned by WordNet so as to more accurately reflect the usage of that term within the organisation. These terms were then recorded within Visio Modeller so that they could be used later to express the organisation's business rules.

4.4 CONCLUSIONS

In this chapter, the research design and how research was conducted has been described. Research design issues were explored by detailing the purpose of the research, the research question, case selection criteria, a description of the context in which the research was conducted and the attempts made to limit the influence of bias. The purpose of the research was to test the hypothesis of this thesis; that ORM constructs can be used to formally express business rules in a form that can still be challenged by end-users. The case itself was chosen because the target organisation exhibited many of the characteristics typical of New Zealand businesses that may suggest that the research findings are applicable in other similar domains. The organisation had also experienced the type of problems with its IT systems that the conceptual approaches advocated within this thesis hoped to alleviate. These problems and the background of the target organisation were then described in detail and included a rationale for selecting the chosen project. Finally, the significance of bias and the attempts to minimise its effect on this research by employing multiple sources of evidence and other techniques were discussed.

The manner in which the research has been conducted was the focus of the second half of this chapter. In this section the data methods employed, which included semi-structured interviews, direct observation of business processes, business documentation, and a lexical database called WordNet, were described. Details of the sampling strategy employed for determining who should be interviewed, how those interviews were planned and how the data collected was validated were also discussed.

CHAPTER 5

ARTICULATING BUSINESS RULES

5.1 INTRODUCTION

The dominant themes discussed in this chapter are those relating to the theoretical issues of this thesis. Thus, Object Role Modelling (ORM) and Business Rules Modelling (BRM) are described in detail and it will be demonstrated how these two modelling approaches may be synthesized into a single conceptual framework for the specification of system requirements.

The Business Rules Group (BRG) has produced a comprehensive taxonomy of business rules (GUIDE Business Rules Project,1997) that defines the structural properties and associated terminology of each category of rule. The model allows analysts to identify business rules within organisations. It also provides a framework for analysts to discuss system requirements with domain experts, using concepts familiar to both parties. However, the BRG have not provided a language with which to express these rules. Although business rules do not present any new concepts unfamiliar to analysts, it will be shown that surprisingly few traditional data modelling tools are able to represent important categories of rules. This chapter explores each category of business rule and compares a number of modelling approaches in their ability to represent these details.

ORM is presented as an alternative data modelling approach and it will be demonstrated that its rich set of constructs allows most categories of business rules to be modelled with little difficulty. It will be shown that in addition to its graphical notation, ORM's linguistic foundation facilitates the verbalisation of business rules. Employing natural language to define system requirements improves the transparency of this activity with the end-user community, which should in turn promote their active involvement in the validation of those requirements. The review of literature in Chapter 2 found that natural languages are often accused of being informal (Gamut 1991), but ORM is built on a strong mathematical foundation (De Troyer & Meersman, 1995) that not only ensures the completeness of each business rule expression, but also allows models to be mapped automatically to a variety of relational schemas.

It will be suggested that as a consequence of ORM's characteristics, it could successfully be employed as a language to express business rules as defined by the BRG. The synthesis of BRM with ORM should provide a useful framework for analysts and domain experts to discuss the system in terms familiar to both. Finally, it will be suggested that the ability of analysts to express business rules to domain experts in natural language, should promote the quality of the resulting system requirements. The review of current literature in Chapter 2 found that a number of industry surveys indicate that about 40% of project failures are related to problems concerning the gathering, specifying and validating of requirements (The Standish Group, 1995), (Conference Board Survey, 2001),(National Audit Office, 2001). If the quality of requirements can be improved, then perhaps the synthesized approach advocated in this thesis may help to address some of the causes of project failure.

Initially, this chapter identifies and provides definitions for the key concepts associated with information systems development (ISD). Once provided, it will allow the conceptual approaches that are the focus of this thesis to be placed into context.

5.2 INFORMATION SYSTEMS DEVELOPMENT

In general terms, ISD can be seen as consisting of two primary activities; analysing business systems and developing computer-based information systems that support them. Hirschheim, Klein & Lyytinen (1995), describes ISD as a set of methods, tools, techniques, and models that are available to assist developers of information systems.

Often, ISD is based on linear process models, one of the most popular being the systems development lifecycle (SDLC) or Waterfall model. The SDLC assumes that developing information systems can be achieved by following a series of interdependent steps in a predefined sequence. Other approaches, such as the Spiral model and prototyping, adopt a cyclical model in which development tasks are performed iteratively in a process of successive refinement.

Associated concepts that are dominant themes in literature concerning ISD are the terms methodology, tools and techniques. Klein & Hirschheim (1991, p157) define a methodology as:

> *An explicit set of assumptions, beliefs and resources (tools and other measures) which guide the analyst from his initial problem definition, to a final conclusion of the project... A methodology not only guides the developer, but allows the project to move through its various stages, each delivering some form of intermediate result. Each stage is further subdivided into specific tasks with which methods and tools are associated.*

Examples of ISD methodologies include Information Engineering (Martin & Finkelstein, 1981) and SSADM (Downs, Clare & Coe, 1988). Both of these methodologies are associated with tools and techniques that analysts employ to perform the specific tasks dictated by the chosen methodology. In the case of SSADM, these include entity relationship (ER) modelling, dataflow diagrams and normalisation. Techniques tend to be more formal than tools in the sense that analysts competent in a technique should produce the same results if they began with the same variables. Consequently, normalisation could be considered a technique. However, quite different results might be expected from different analysts when constructing an ER model for a given scenario. As such, ER diagramming could be considered a tool, rather than a technique.

5.2.1 Prescriptive and Descriptive Modelling Approaches

Differences in the results obtained by analysts may arise from the descriptive nature of ER modelling. Atkins (1996) defines descriptive modelling procedures as those that recognise the existence of a subjective reality that requires imagination, creativity and experience to represent that reality within models. Thus, the skills associated with descriptive approaches, such as ER modelling, are acquired by exposure to many examples of such models, rather than through a familiarity with a predefined set of

rules and steps that are followed in a given sequence. Consequently, analysts have no mechanism for determining the most appropriate ER model for a given situation or even to recognise such models when constructed (Simsion, 1994).

Prescriptive modelling approaches adopt a different perspective in that reality is considered to be objective. As such, the opportunity exists to discover this objective reality in sufficient detail so as to represent it as a complete model (Atkins, 1996). Prescriptive techniques are associated with more detailed steps and guidelines than descriptive approaches. Following these steps, analysts are more likely to construct similar models for a given situation than those analysts adopting descriptive modelling approaches.

5.2.2 ORM and BRM in the Context of ISD

Using the above definitions, Object Role Modelling (ORM) can be best described as a prescriptive data modelling technique. The technique itself is associated with the Conceptual Schema Design Procedure (CSDP), which defines in detail the steps required to construct an ORM conceptual schema. However, CSDP cannot be considered a methodology since it does not encompass all of the tasks associated with ISD. For example, analysts need to apply other tools and techniques to define the processing requirements of the system.

The second conceptual modelling approach explored in this Chapter is Business Rules Modelling (BRM). The Business Rules Group (BRG) have defined a comprehensive taxonomy of business rules that allow analysts to clearly determine what is and what is not a business rule and to categorise those rules into specific types (GUIDE Business

Rules Project, 1997). In general terms, the BRG view rules as assertions that dictate or least influence the structure and behaviour of organisations. The BRG have also described the structure of business rules themselves as a conceptual model in the form of an ER diagram. However, the BRG have not provided detailed guidelines as to how analysts discover business rules or any formalism that may be used to articulate these rules, once they have been found. In the absence of such information, BRM could be categorised as a descriptive modelling tool for defining the structure and behaviour of organisations.

ORM is a technique that is applied in the requirement analysis stage of information systems development. Although, employing Visio Modeller, ORM's automated tool support, analysts are able to generate databases derived from ORM diagrams. However, the major deliverable associated with the application of ORM is the production of a conceptual schema and the CSDP technique describes in detail the steps required to achieve this goal. The generation of the database itself is performed automatically by Visio Modeller by mapping the conceptual schema into an internal schema for the target DBMS.

Modelling business rules can be considered as a tool that is applied during requirements analysis stage of ISD. Requirements analysis consists of two steps involving requirements acquisition and requirements modelling (Greenspan, Mylopoulos & Borgida, 1994). Vitalari (1992) has suggested that all requirements are initially propositions or assertions about the desired information system that should be tested before they become part of the final specification. Since business rules describe

the structure and behaviour of organisations as a series of assertions, BRM could provide a useful framework for identification and testing of these assertions.

In summary, it is suggested that ORM is a prescriptive data modelling technique for defining a conceptual schema during the requirements analysis stage of ISD. BRM, however, is a descriptive modelling tool for defining the structure and behaviour of organisations, which is also applied during the requirements analysis phase of ISD. Since business rules are assertions about the organisation, there are opportunities to test these assertions before they become incorporated into the specification of requirements.

5.3 DEFINING BUSINESS RULES

As yet, there is no generally agreed definition as to what actually constitutes a business rule. This uncertainty manifests itself by the diversity of opinion on this issue. Sandy (1994)notes that this opinion ranges from the identification of critical success factors and quality goals at one extreme, to a detailed description of database integrity constraints at the other. The work of the Business Rules Group (BRG), despite the fact that it originally consisted of a number of notable members including E.F. Codd and Charles Bachman, has apparently done little to change this current state of affairs. However, the project they undertook in 1997 (GUIDE Business Rules Project, 1997) that sought to define business rules, is still one of the most comprehensive studies of its type at the time that the literature review of this thesis was conducted. This provided the rationale for accepting the findings of this project

and for incorporating their definitions into a synthesized conceptual model for the specification of system requirements.

The report produced by the BRG describes a business rule as "a statement that defines or constrains some aspect of the business. It is intended to assert business structure or to control or influence the behaviour of the business" (GUIDE Business Rules Project 1997, p.4-5). Since the focus of the group's work was on business rules from an information systems perspective, these statements or assertions that define business structure and behaviour concern the data recorded by system and the constraints on the values of that data.

Petri-nets and finite state diagrams are often used to model behaviour in information systems. However, behaviour from the context of the definition provided above does not have the same meaning. Petri-nets, for example, depict states and transitions (events) and the dependencies between them (Peterson, 1981), whereas finite state diagrams show states and their interdependencies (Davies, 1988). Modelling behaviour in this context seems to be restricted to modelling time-dependant behaviour. This emphasis on time related events, a general theme of many behaviour modelling tools and techniques, could make the capture of more static, structural rules difficult to accomplish. In contrast, the BRG have dealt exclusively with issues relating to information structure. Consequently, they have not had to specifically address the role of events. In their report (GUIDE Business Rules Project, 1997, p.5), they state,

From the perspective of an information system, an event only manifests itself by virtue of the fact that persistent data has come into existence or has been modified. The business rules described here control whether or not such data maybe created or changed and the implications when this occurs.

Behaviour then, from the perspective of the BRG, concerns the constraints on data and the influence this has in prescribing the organisation's activities.

5.4 BUSINESS RULE CATEGORIES

The BRG have defined 4 major categories of business rules that are briefly described in the following section. Once introduced, the rationale for modelling rules is described and is followed by a detailed examination of each rule category.

5.4.1 Definitions of Business Terms

The terminology applied within business organisations has been identified as a category of business rule. This reflects the fundamental importance of language in the articulation of business rules. Concepts and the shared meanings adopted by organisations must therefore be captured before these terms can be related to one another to express other categories of business rules. Using traditional analysis tools, these terms are identified by the names given to entities in ER type diagrams or classes in Class diagrams.

In the case study described in Chapter 4, WordNet, a lexical database, was used to define generic terminology applied within the target organisation. Terms specific to the organisation were defined using a combination of internal and external

documentation and glossaries. However, it was also important that management and employees validated these terms so that a final consensus could be reached before they were used to express other types of business rules.

5.4.2 Structural Assertions

The second category of business rules defined by the BRG identifies rules that describe the structure of an organisation through assertions that relate terms to one another. Such rules are called structural assertions and are most often represented graphically in traditional approaches to ISD as classes or entities involved in relationships with other classes or entities. The approach taken in the case study of this thesis was to express structural assertions both graphically and as natural language sentences using Object Role Modelling constructs. This approach is described in more detail in sections 5.6.3 and 5.6.4.

5.4.3 Action Assertions

The third category of business rules are those rules used to define the behaviour of the organisation in terms of the constraints under which they operate. They are called action assertions and concern dynamic aspects of the organisation and thus are more likely to change over time than rules that define organisational structure. This is because these rules reflect business policy that often evolves in response to environmental pressures. Action assertions are closely related to the constraints on data since these effectively dictate what actions can take place. Such rules can be initially expressed as natural language statements. This was the approach taken in the case study of this thesis by applying ORM constructs to express these rules, thus allowing domain experts to more actively participate in their validation.

5.4.4 Derivations

The final category of business rule concerns how new data is derived from existing data. For example, a derivation might describe how an invoice total is derived from individual line totals. Derivations are expressed either as mathematical algorithms, or as inferred logical inductions or deductions. In both situations, a derived fact is produced as a consequence of applying a mathematical calculation or inference and may be specified using a natural language such as Formal Object Role Modelling Language (FORML).

5.5 WHY MODEL BUSINESS RULES?

As the discussion above suggests, business rules do not in themselves present any new concepts that are not already familiar to analysts. However, business rules implemented using traditional approaches to ISD often leads to rules that are 'scattered' throughout the information system. For example, the cardinality of a relationship between two entities are details found within the database schema, whereas as the algorithm to calculate an employees sales commission would be found as a section of code in one of the information system's programs. Since business rules are typically not centrally located, modifying them can be challenging. First, the rule has to be found, which could be difficult if that rule is embedded within code somewhere in the system's many programs. Modifying the rule could also lead to unforeseen consequences in other parts of the system.

Analysing and recording rules separately from other system details can avoid both of the problems highlighted above. Constructing a repository of rules provides a central location for all rules allowing them to be managed much more efficiently. This is particularly the case for Action Assertion and Derivation rule types, since these rules tend to be more dynamic in nature than the more static Structural Assertions. When business rules evolve and change in response to the organisation's environment, a central repository of rules provides a mechanism to quickly locate that rule and modify it accordingly. Software applications known as rule repositories (sometimes referred to as a business rules engines) have now become commercially available that also provide impact analysis tools with which to predict effects of modifying business rules. Finally, since each rule is recorded only once in a repository, modifying a rule will cause the effect of that rule to change accordingly, wherever that rule is invoked.

Perhaps even more significant than the pragmatic considerations discussed above, is that business rules provides a conceptual framework with which to discuss the organisation with domain experts. For example, analysts will often use an ER type diagram to document the data requirements of a system. However, this data orientated abstraction of the real world could be difficult for many domain experts to validate unless they are already familiar with such tools. Since business rules originate as policy statements issued by management, a rules-based focus during requirements analysis should improve the transparency of this activity for domain experts. Finally, since all business rules can be expressed as assertions, it should be possible to articulate rules as natural language statements facilitating more effective end-user validation.

5.6 EXPRESSING BUSINESS RULES USING OBJECT ROLE MODELLING

The Business Rules Group (BRG) has proposed a conceptual model of business rules, the details of which are shown in Appendix 10. This model decomposes the business rule types identified in sections 5.4.1 to 5.4.4 into their constituent elements so that analysts can clearly identify each rule type and express them completely. However, the BRG has not proposed a language by which these rules can be expressed. Thus it will be demonstrated how Object Role Modelling (ORM) constructs may be applied to articulate all forms of rules identified by the BRG. In doing so, the primary focus of this thesis is addressed by determining the efficacy of the approach for the specification of business rules[1]. It will also be demonstrated that most other traditional tools and techniques could not achieve the same goal. They simply lack the necessary constructs to model the subtle, but important system details that business rules attempt to capture (ter Hofstede, Proper & van der Weide, 1994). Finally, it will be described how these rules may be verbalised using ORM's formal natural language (FORML). It will be argued that this ability to express all categories of rules in a format comprehensible to domain experts, should allow these people to actively challenge the perceptions of the analyst, thus facilitating more effective validation.

5.6.1 Defining Business Terminology

A preliminary activity that precedes the formulation of other types of business rules concerns the definition of terms used within the organisation. As previously stated in section 5.4.1, these terms are the fundamental 'building blocks' of all other categories

[1] Refer to section 4.2.2 for a description of the research question of this thesis.

of business rules. As such, it is important that a consensus is reached as to the precise meaning of the terms employed within the organisation. The methodology associated with ORM, Conceptual Schema Design Procedure (CSDP), does not specifically identify the definition of terms as a required step. However, ORM's automated tool, Visio Modeller, does provide a feature that allows such details to be recorded as a note that annotates each object identified within an ORM model.

The approach adopted in this thesis was to use a combination of internal and external documents, together with WordNet[2], a lexical database, to identify and define a preliminary list of business terminology used within the target organisation. Having constructed a preliminary glossary, these details can then be validated by domain experts and modified accordingly until a general consensus on content and meaning has been reached.

Each defined term is itself a business rule in the sense that each definition is an assertion about some aspect of the organisation. Terms were therefore defined using a general notation that was applied to each definition, in the form:

<Object> is a ....

For example, Qualification has been defined as follows:

> Qualification is a set of requirements for certification established by a recognised standards-setting body or an education provider.

As can be seen in the above example, each definition can identify other terms that may also need to be defined. For instance, 'certification', 'standards-setting body' and 'education provider' could all be terms that require definition. The decision to provide or not to provide a definition of a business term was dictated by whether or

[2] Refer to section 4.3.9 for a discussion on how WordNet was employed in this thesis.

not that term itself was used to express other kinds of business rules. Only when a term was used to express other rules was its definition considered appropriate.

5.6.2 The Structure of Structural Assertions

As previously stated in section 5.4.2, Structural Assertions define static, structural details of the organisation by associating terms together to form an assertion. Using the IDEF1X notation, the BRG have produced a conceptual model that describes the structural elements of this business rule type (GUIDE Business Rules Project, 1997, p.15) which has been reproduced below.

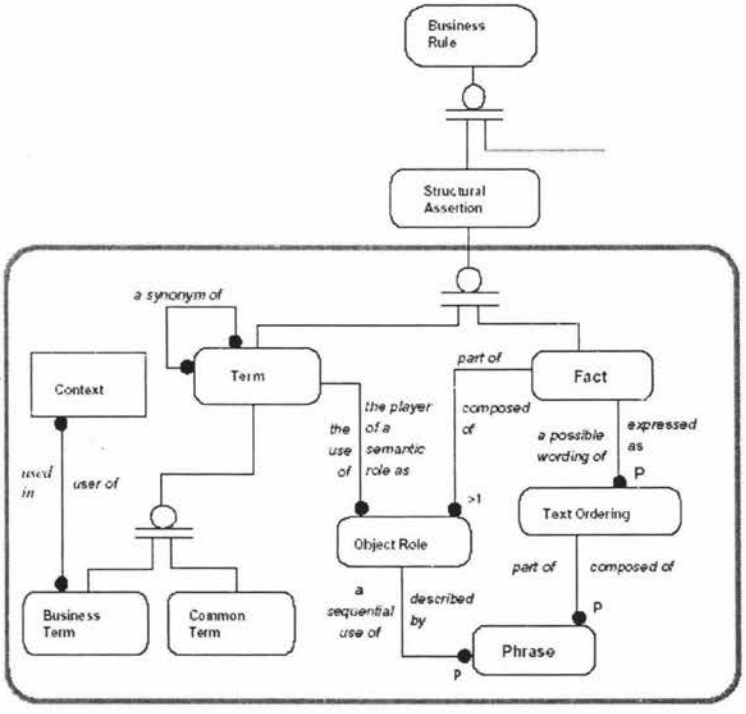


Figure 5.1 – The Composition of Structural Assertions[3]

The first detail to note in the above model is that there are two kinds of Structural Assertion; Terms and Facts. Terms themselves can be either Business Terms or Common Terms. Common Terms are simply words used in everyday language, but Business Terms have a specific meaning in at least one business Context.

[3] Reproduced from GUIDE – Business Rules Project, Final Report, 1997, p.15

A Fact is an assertion of an association between two or more Terms and each Term may be used in one or more Facts. Facts may be Derived or Base Facts, although that distinction is not indicated in Figure 5.1. Derived Facts are discussed in detail in section 5.8. The remainder of this section concerns Base Facts.

Base Facts do not have to be a simple binary association between two Terms. The conceptual model shown in Figure 5.1 recognises the need to express compound associations in a natural format, but this flexibility assumes that the language used to articulate such rules is capable of expressing associations of any arity. This of course is not true of many types of ER diagram and highlights the fact that numerous modelling tools and techniques reflect their underlying implementation technology. However, this is not the case with ORM and it will be demonstrated in section 5.6.3 how this language can conceptually express complex associations between Terms.

Figure 5.1 shows that each Fact must consist of two or more Object Roles where each Object Role depicts a semantic role played by a Term within a Fact. In the example below, there are two Object Roles associated with the Fact that there is an association between the Terms 'Student' and 'Enrolment'.

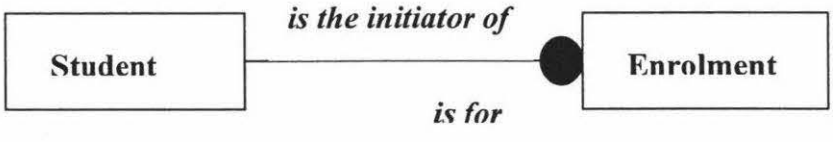


Figure 5.2 – Associating Terms using IDEF1X

One Object Role asserts that the Term 'Student' plays a semantic role in a Fact and the other Object Role is that "Enrolment' also plays a semantic role in the same Fact.

Even binary associations between Terms may be used to express a Fact in a number of ways. Using the above example, the fact that there is an association between the Terms 'Student' and 'Enrolment' can be expressed in two ways, one for each direction in which the association is verbalised, as demonstrated below.

> Each Student may be *the initiator of* one or more Enrolments.
>
> Each Enrolment must be *for* one and only one Student.

Thus, Facts may be expressed as one or more Text Orderings, as shown in Figure 5.1. In the example above, each sentence about 'Student' and 'Enrolment' is a Text Ordering of the Fact associating these two Terms. Each Text Ordering must be composed of one or more Phrases where each Phrase must be a sequential use of one of the Object Roles that is part of the Fact. This means that the Phrase specifies the sequence of the Object Role in the Text Ordering. Using the above example, the Phrase 'Each Student' describes the Object Role (i.e. 'Student') as the subject in this Text Ordering. In the second Text Ordering of the same Fact, the Term 'Student' appears in a different Object Role that defines it as the object of 'Enrolment'.

5.6.3 Using ORM to Express Structural Assertions

Although the IDEF1X notation has been used in the previous section to describe the structure of Structural Assertions, in this section Object Role Modelling is introduced and will be used to demonstrate how rules of this type can be formally expressed. The rationale for employing this technique has been discussed in section 2.4 of Chapter 2 However, the points raised in that discussion will also be highlighted here where appropriate.

When applying Object Role Modelling (ORM), the organisation is described in terms of objects playing roles. Statements that describe these associations are elementary (atomic) facts. These facts are assertions that particular objects play particular roles.

Objects can exist in one of two basic forms; Entities types or Values types. Value types can either be character strings or numbers and are identified by constants. Character strings are shown inside single quotes and numbers, without quotes. ORM makes no use of attributes during the analysis stage, although Value type objects often become attributes when the conceptual schema is mapped onto a relational schema. The rationale for this approach is that during analysis, what was originally modelled as an attribute can later be re-defined as an entity as more details regarding to domain are discovered. Therefore, by not using attributes during analysis, models are more stable and simple (Halpin, 2001). However, the price paid for this stability and simplicity is the tendency for ORM graphical representations of the business domain to appear 'cluttered' and possibly confusing to those not familiar with its notation.

The term entity has a more conceptual meaning within ORM than it does in most other modelling tools and techniques. In ORM, an entity is simply a described object. An entity is described in terms of it entity type, reference mode and value. Entity type is the name of the set of all possible instances of that entity type. The reference mode is the means by which a single instance of that entity type can be uniquely referenced and value is the value of that instance. Note that Value type objects do not possess a reference mode since their instances are literal constants and so they identify themselves. The sentence below illustrates these details:

A Car with registration number 'XYZ 123' has a Colour of 'Red'.

In this example, the following components can be identified:

Entity Type: Car
Value Type: Colour
Reference Mode: registration number
Value: 'XYZ 123', 'Red'

Often, sentences are abbreviated to express facts more succinctly using the syntax of Formal Object Role Modelling Language (FORML). Using the example above, the same fact may be written in FORML as follows:

Car (regNr) 'XYZ 123' has Colour() 'Red'.

Note that Colour is a Value type object and therefore has no reference mode indicated between the brackets and the reference mode of Car has been abbreviated to regNr.

In order to specify the role(s) that each object plays, ORM employs logical predicates. In general, an *n*-ary predicate is a sentence with *n* object holes where *n* is the arity of the predicate. Since order is significant, *n*-ary predicates have a sequence of *n* object terms. The example below illustrates a quaternary predicate (i.e. has four object holes).

Client (cltCode) 100 **on** Date (dmy) '05/12/03' **hires** Car (regNr) 'XYZ 123' **for** Charge() $100.

In this example, the sentence uses the predicate ".. on .. hires .. for .. ". In general, entity types, value types and reference modes appear as nouns and predicates as verb

phrases. Predicates may also be reversed to allow an inverse reading of the same fact, as illustrated in the binary predicate below.

Employee (empcode) 200 **is employed by / employs** Department (deptcode) 'Accounts'.

The standard left to right reading of the above sentence involves the predicate on the left of the slash '/' and the inverse reading involves the predicate on the right.
It should be clear from the above discussion that there are many similarities in the underlying concepts of ORM and BRM. Assertions about an organisation are expressed as sentences in both cases, although each is associated with different terminology to describe the structural elements of these sentences. These differences in terminology are highlighted below.

Table 5.1 – Terminology Used in ORM and the BRG

BRG Terminology	ORM Terminology	Common Meaning
Term	Object (Entity or Value)	Thing of interest described by a word or phrase.
Fact	Elementary Fact	An assertion associating two or more Terms/Objects.
Object Role	Role	Semantic role played by a Term/Object in a Fact.
Text Ordering	Sequence	One possible wording of a Fact

5.6.4 Graphical Representation of Structural Assertions

Not only does ORM allow business rules to be articulated as formal sentences, but it also provides a graphical notation to represent these rules. When employing Visio Modeller, ORM's automated tool support, graphical models can be constructed automatically if business rules are entered in FORML.

Using the example fact:

'Client (cltCode) 100 **on** Date (dmy) '05/12/03' **hires** Car (regNr) 'XYZ 123' **for** Charge() $100',

the following model may be drawn:

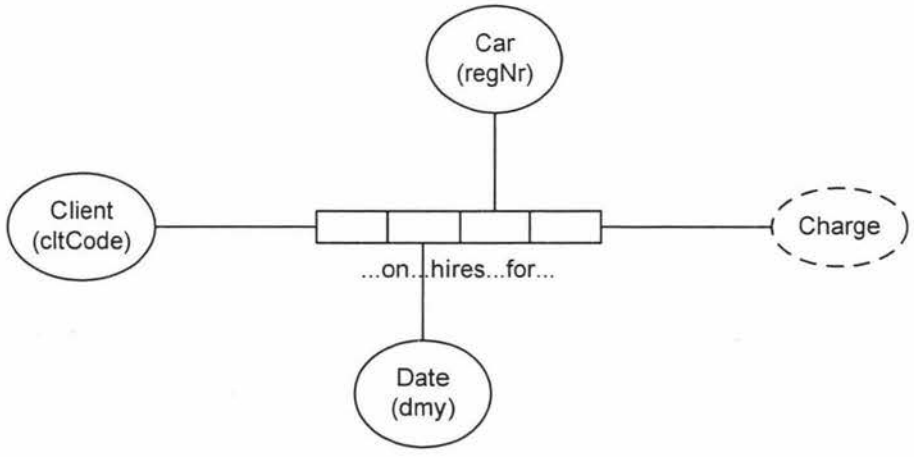


Figure 5.3 – Graphical Representation of a Structural Assertion

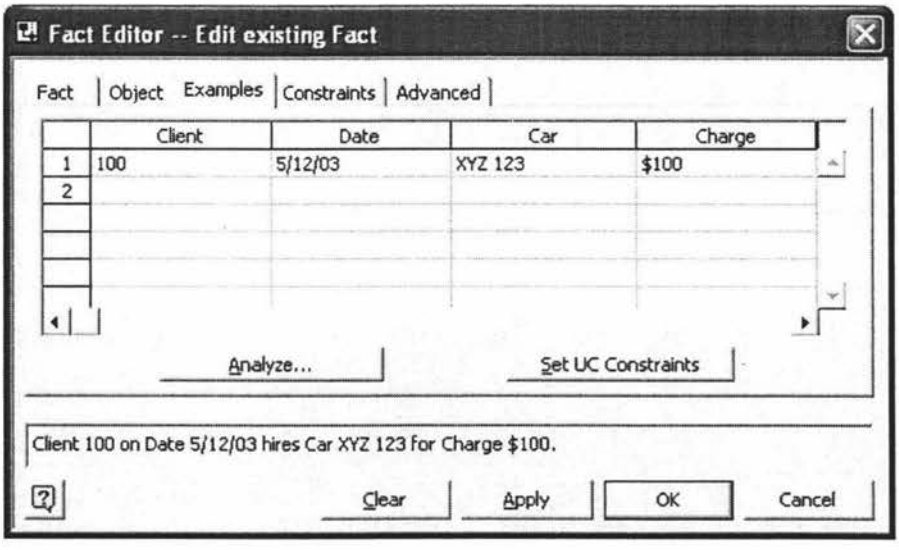
The model depicts the above fact in its general form, i.e. the values of the specific instances of each object are not shown. However, these details can be provided in the form of fact tables, the details of which are used by Visio Modeller to define uniqueness constraints i.e. whether or not the values of instances associated with each object can be repeated within the same column of a Fact table. Uniqueness constraints are discussed in detail in section 5.7.3

Objects, both Entity and Value type, are depicted as ovals in figure 5.3. Value type objects are shown using a dashed line and with no reference scheme indicated. The predicate ".. on .. hires .. for .. " is shown adjacent the role sequence (an ordered list of roles) for this fact. The role sequence is represented by the rectangle, which is sub-divided into 4 role boxes indicating a quaternary predicate. The line extending from each of these role boxes terminates at the Object with which that role is associated.

The order of the role boxes defines the order in which each associated Object is placed in the object holes of the predicate ".. on .. hires .. for ..". In this example, the Object 'Client' is placed in the first object hole (indicated by the ellipses) of the predicate and 'Date' in the second etc. Thus, the order of the role boxes defines the text ordering of the Fact.

If example data is provided in the form of a fact table, each role box is associated with a sample population of instances from one of the fact table's columns. Thus, each instance in one of the fact table's columns depicts an instance of an Object playing a specific role in a Fact. Using the above example, the following fact table could be provided:

Table 5.2 – Example Fact Table



Each column in the above table corresponds to a population associated with each of the role boxes shown in Figure 5.3 e.g. the Client column is associated with the first role box and Date, with the second etc. Note also how Visio verbalises the Fact using the data drawn from this table, thus further facilitating end-user validation of rules by using actual examples extracted from the business domain.

5.6.5 Fact Classification

The Business Rules Group (BRG) classifies Facts into one of three types depending on the manner in which the Terms associated with a Fact are employed. The Terms used to express a Fact may be an Attribute of another Term, or a Term may be used to express a Fact that associates one or more Terms together (known as Participation) in the form of a relationship. Finally, a Term may be a Generalisation (supertype) of one or more other Terms (its subtypes). This classification scheme for Facts is illustrated below.

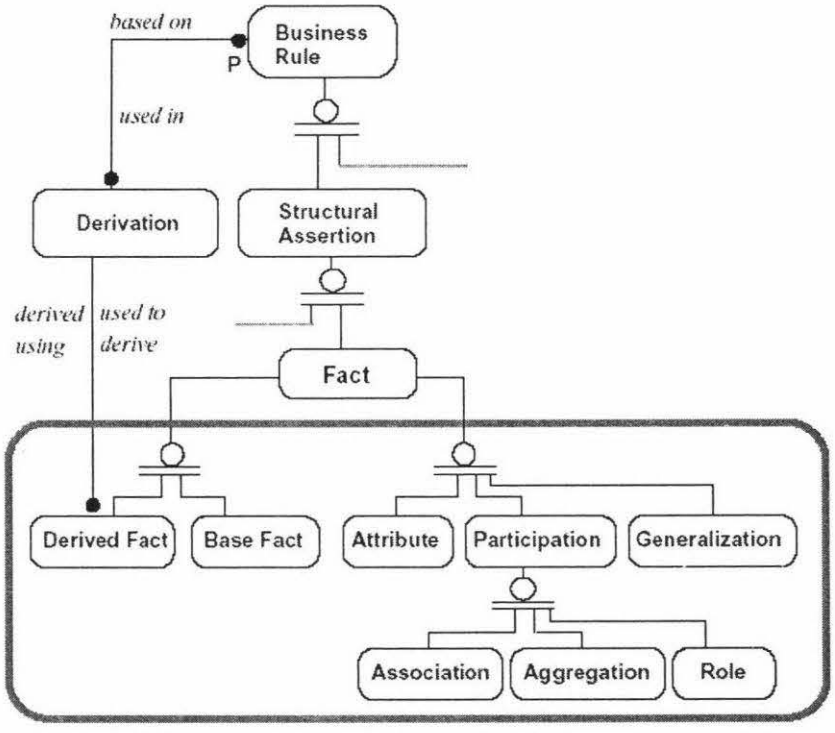


Figure 5.4 – The BRG's Classification Scheme of Facts

ORM does not initially make a distinction between entities and attributes in order to foster conceptually simple, stable models. What would be thought of as an attribute with ORM is simply a Value type Object that participates in a role with at least one other Object. Thus, Facts classified as Attribute can be considered as the equivalent of the BRG's Participation type Facts.

As can be seen in figure 5.4, Participation type Facts are further classified as being Association, Aggregation or Role. Aggregation describes those Facts that assert that one Term consists of an aggregation of another Term. An example used in section 5.6.3 and reproduced below is an example of such an Aggregation type Fact.

Employee (empcode) 200 **is employed by / employs** Department (deptcode) 'Accounts'.

In this example, a Department employs (or consists of) Employees.

Role type Participation Facts describe an association that exist between Terms. Again, using a previous example from section 5.6.3, Role type Participation can be illustrated, as shown below.

Client (cltCode) 100 **on** Date (dmy) '05/12/03' **hires** Car (regNr) 'XYZ 123' **for** Charge() $100.

In this example, the predicate '.. on .. hires .. for ..' describes the roles that associates the Terms 'Client', 'Date', 'Car' and 'Charge' within a Fact.

Association type Participation Facts describe an association between Terms that do not fall into either the Role or Aggregation category. The BRG provide two examples of Facts that fall into this category and are reproduced below.

> 'Car models may be requested by customers'
> 'Cars may be rented by customers'
> (GUIDE Business Rules Project, 1997, p.22)

Within ORM, such facts are described simply as roles between objects. For example, the first fact would be articulated as follows.

Car Models are requested by / requests Customer

Uniqueness and Mandatory Role Constraints would then be applied to this Fact as appropriate to the scenario being modelled. These two constraints are described fully in section 5.7.3 to 5.7.6.

The final category of Fact that has been identified by the BRG concerns Generalisation. In order to capture such facts within the business domain, the data modelling tool or technique employed must support subtyping. This issue is discussed in detail in the next section.

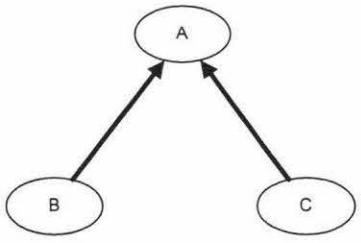
5.6.6 Subtyping

Subtyping concerns the classification of objects into one or more specific types. The main purpose for introducing these structures into an ORM data model is to allow other constraints to be applied against a specific role played by only one subtype. Subtyping also facilitates the re-use of components within the model that can significantly reduce code duplication. Finally, subtyping can also be used to describe classification schemes that may exist within the organisation.

ORM constructs allow a variety of subtyping concepts to be conveyed both graphically and verbally. The examples below detail these constructs with their corresponding verbalisations.

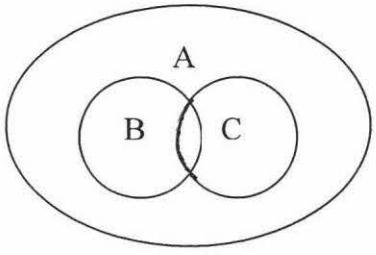
Figure 5.5 - Overlapping Subtypes

ORM Representation



Each A is both B and C

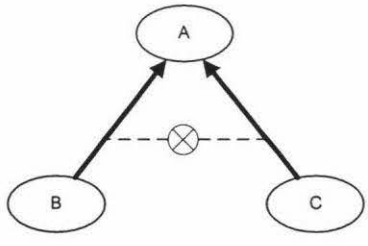
Euler Representation



In figure 5.5, the population of the types may overlap as shown in the corresponding Euler diagram on the right. In figure 5.6, both B and C are mutually exclusive so that no A is both B and C. Figure 5.7 illustrates a situation in which the subtypes B and C exhaust A i.e. Supertype A is the union of both B and C. Finally, figure 5.8 represents a combination of both exhaustive and mutually exclusive subtypes.

Figure 5.6 – Mutually Exclusive Subtypes

ORM Representation



No A is both B and C

Euler Representation

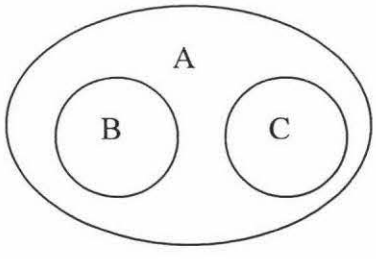
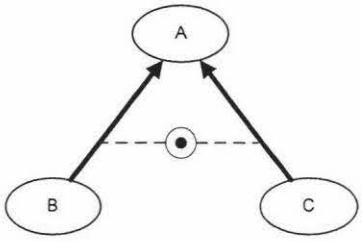
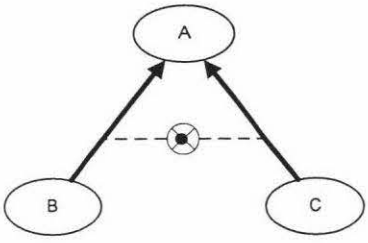


Figure 5.7 – Exhaustive Subtypes



Each A is at least one of B or C

Figure 5.8 – Exhaustive and Mutually Exclusive Subtypes



Each A is exactly one of B or C

As a practical example of some of these concepts, the model in figure 5.9 describes the situation within a typical university in relation to General and Academic staff, their association to a department and the papers offered by that department.

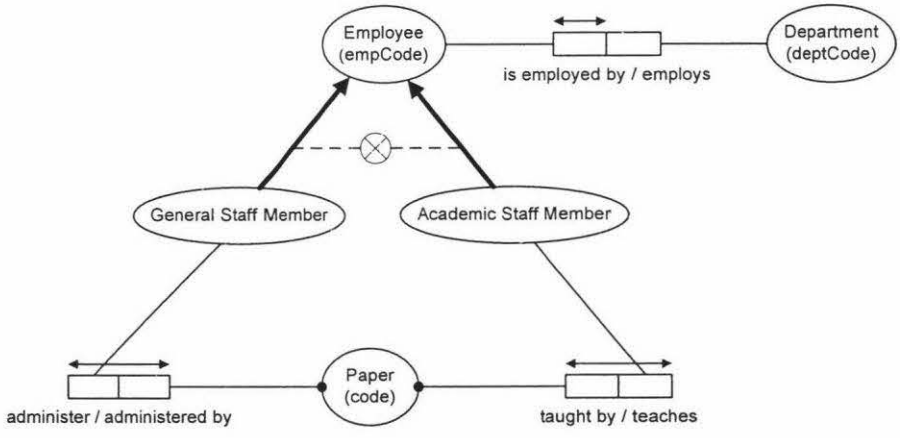


Figure 5.9 Applying Subtyping Concepts

General Staff and Academic Staff Members are subtypes of the Supertype named Employee. The exclusion symbol ⊗ between the two sub-types indicates that no employee is both General Staff and Academic Staff. However, these subtypes do not exhaust the supertype Employee, since other types of employee such as canteen or grounds staff are also employed by most universities. Also note that although both

subtypes have an association with 'Paper', each plays a different semantic role with that object. Subtyping in this context is a common approach for distinguishing between the roles played by different subtypes. Finally, ORM allows subtype/supertype structures to be verbalised using Formal Object Role Modelling Language (FORML) constructs. The example in Figure 5.9 would be verbalised as follows.

'Each General Staff Member and Academic Staff Member is an Employee, but not vice-versa'

5.6.7 Subtyping in Other Modelling Approaches

Barker's notation is a commonly used ER diagramming tool that has been adopted by the Oracle Corporation for use within its Case*Method approach for systems development (Barker, 1990). Although popular, it lacks the richness of constructs to represent any but the simplest of subtyping concepts. As such, this notation can only represent the concepts of Exclusion and Exhaustion. More complex, but still common, overlapping subtypes cannot be represented. Using the example illustrated in figure 5.9, Barker's notation would depict the supertype/subtype details in this scenario as follows.

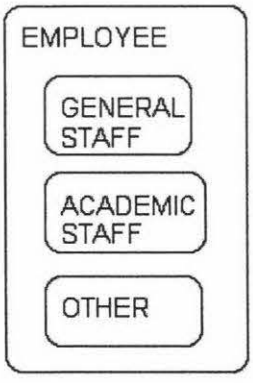


Figure 5.10 – Subtyping using Barker's Notation

In this example, the subtypes of General Staff and Academic Staff are represented by rectangles placed within the supertype of Employee. Each subtype is mutually exclusive (overlapping subtypes are not allowed in Barker's notation, but not required in this example) and the subtypes must exhaust the supertype Employee. Since other types of employee may exist (groundsmen, canteen staff etc), the subtype 'Other' is included. Within ORM there is no need to include the subtype 'Other', unless the subtype plays a specific role. Since Barker's notation lacks a linguistic framework, there is no formal way to verbalise the above structure.

Within UML, the concepts of supertype and subtype are referred to as superclass and subclass, respectively. But where subtypes in ORM inherit the roles of the supertype, in UML, subclasses inherit the attributes, associations and methods of the superclass. However, this discussion will focus UML's data modelling characteristics (attributes and associations) and ignore aspects related to behaviour (methods).

Unlike many ER modelling approaches, including Barker's notation, UML allows a variety of supertype/subtype characteristics to be modelled (Rumbaugh, Jacobson & Booch, 1999). Figure 5.11 compares ORM and UML in their representations of subtypes.

UML Representation **ORM Representation**

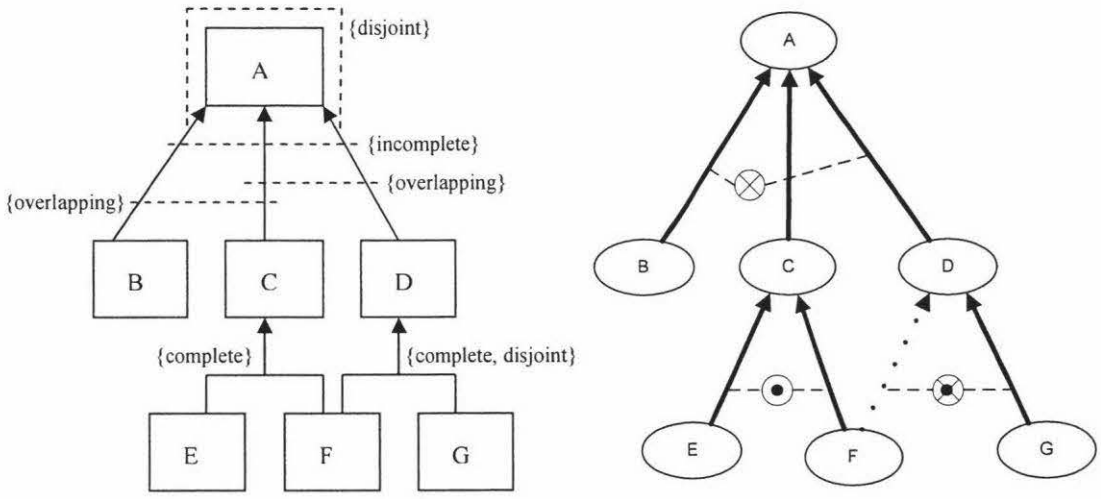


Figure 5.11 – Comparison of ORM's and UML's Subtyping Constructs

UML includes four constraints that are used to indicate whether subtypes are exhaustive ('complete' and 'incomplete') and exclusive ('disjoint' and 'overlapping'). Subtypes that collectively exhaust the supertype are 'complete', as indicated for the subtypes 'E' and 'F', which exhaust the supertype C. This is also the case for subtypes 'F' and 'G', which collectively exhaust the supertype 'D'. Where subtypes do not exhaust the supertype, the keyword 'incomplete' is used together with a dashed line that links the subtypes in question. This is indicated in figure 5.11 for the subtypes 'B', 'C' and 'D'.

In UML, subtypes in which the populations overlap are indicated by a dashed line that connects the subtype links together with the keyword 'overlapping'. In figure 5.11, the populations of subtypes 'B' and 'C' overlap. This is also true for subtypes 'C' and 'D'. Finally, when subtypes are mutually exclusive in the sense that their populations must never overlap, this fact is indicated with a dashed line connecting the subtypes in question and the keyword 'disjoint' is used to label that line.

By comparison, ORM uses a circled dot to indicate that subtypes are exhaustive and a circled 'X' to indicate that subtypes are mutually exclusive. If subtypes are both exhaustive and exclusive, a circled, crossed dot symbol is used. Subtypes overlap by default in ORM and so no symbol is required to indicate this constraint. Note that both ORM and UML support multiple-inheritance, as illustrated by subtype 'F' in figure 5.11. ORM uses a dashed-line to connect the subtype to the supertype to indicate multiple-inheritance and to distinguish between cases of single-inheritance where a solid line is used. Inheritance of operations and methods is a feature of UML that facilitates the modelling of behaviour. ORM, being strictly a tool for modelling data, possesses no comparable construct.

5.6.8 Conclusions Regarding ORM and Structural Assertions

In the above discussion, ORM constructs have been employed to express Structural Assertion type business rules, as defined by the Business Rules Group (BRG). It has been demonstrated that not only can business rules of this type be expressed graphically, but can also be described in the restricted formal grammar of Formal Object Role Modelling Language (FORML). Despite the mathematical foundations upon which this language is based, its ability to verbalise the static, structural details concerning the domain should allow end-users to participate in the validation of these details. Although all traditional data modelling tools possess constructs that enable the capture of these structural details, very few allow these details to be verbalised easily since, unlike ORM, most do not possess a linguistic framework. Without such a framework, one would expect that the ability of domain experts to challenge the understanding of analysts modelling their domain to be hindered.

5.7 ACTION ASSERTIONS

As previously stated in section 5.4.3, Action Assertions are business rules that define the behaviour of the organisation in terms of the constraints under which it operates. The Business Rules Group describe this business rule type as the "constraints on results that actions can produce" and concerns "dynamic aspects of the business" (GUIDE Business Rules Project, 1997, p.25). This focus on the dynamic nature of an organisation suggests that Action Assertions are a category of business rules that are more likely to change over time. Thus, the ability of analysts to capture these rules and subsequently modify them should be an important consideration in systems development. However, after a review of current literature, it was discovered that very few modelling tools and techniques are capable of modelling the subtle, but common features this rule type. The following sections describe action assertions, as defined by the BRG, and demonstrate how ORM and other modelling tools and techniques attempt to model these details.

5.7.1 Action Assertion Classification

The BRG has proposed that 3 classes of Action Assertion exist; Condition, Integrity Constraint and Authorisation, as shown in figure 5.12. A Condition is an assertion that if something is true, then some other business rule will be applied. An Integrity Constraint is an assertion that must always be true and as such, would prohibit any actions that would violate such rules. Finally, an Authorisation defines what actions may be performed by individuals, departments or machines.

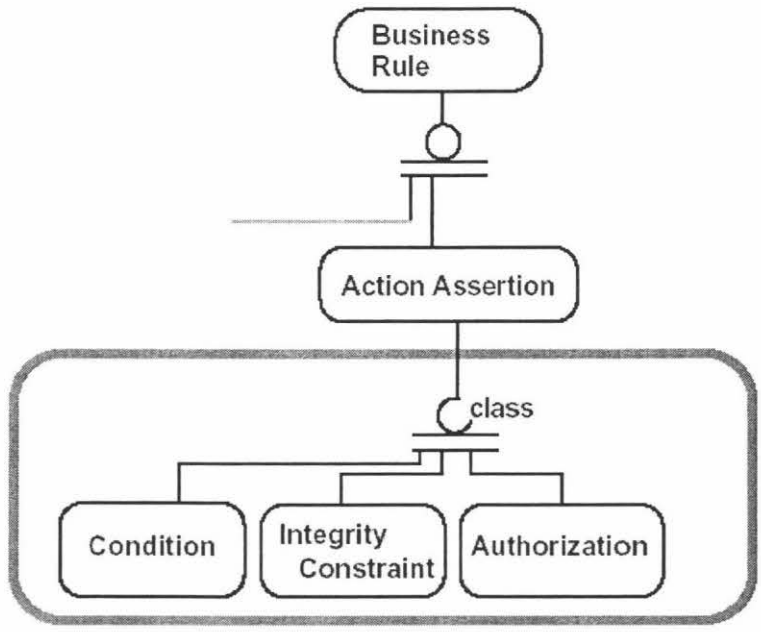


Figure 5.12 – Action Assertion Classification[4]

In sections 5.7.2 to 5.7.21, these classes of Action Assertion are discussed. It will be demonstrated how ORM constructs allow such business rules to be captured as well as highlighting the difficulty that most traditional data modelling tools have in representing these details.

5.7.2 Integrity Constraints

As previously stated, this class of Action Assertion concerns those business rules that must always be true and most commonly appear on ER type diagrams as the cardinality and optionality of relationships. However, integrity constraints can also include set comparisons between populations, which very few data modelling tools and techniques facilitate. This inability confirms the findings of ter Hofstede, Proper et al., (1994) who suggest that the constraints that apply to data structures cannot be easily represented, if at all, in most data modelling techniques. Not only can ORM

[4] Reproduced from GUIDE – Business Rules Project, Final Report, 1997, p.27

capture these details but the approach also allows these constraints to be verbalised easily, thus providing opportunities for domain experts to participate in their validation. Unlike ORM, most other data modelling tools do not possess a linguistic framework and thus have difficulty in expressing constraints naturally. This problem becomes evident when even basic constraints such as the cardinality of a relationship is verbalised. For example, very few data modelling tools and techniques facilitate an inverse-reading of a relationship since most require only a single word or phrase to describe an association between entities. Barker's and the IDEF1X notation are two of the few ER type diagramming tools that do not suffer from this limitation.

5.7.3 Cardinality

Within ORM, cardinality is referred to as a uniqueness constraint and concerns the uniqueness of instances that populate the columns of the fact tables. Using a previous example, a uniqueness constraint (shown as a double-headed arrow) has been placed over the role box 'is employed', as shown below.

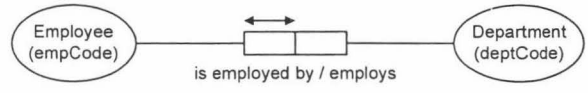


Figure 5.13 – A Uniqueness Constraint on a Binary Fact Type

The uniqueness constraint over that role box indicates that for all possible populations, instances of that role's fact column (i.e. Employee) must be unique, as illustrated below.

	Employee	Department
1	100	Accounts
2	200	Accounts
3	300	Sales

Table 5.3 – Fact Table Corresponding to the Uniqueness Constraint

Since each department, in this example, can employ more than employee, instances in Department column of the above fact table are not unique and thus no uniqueness constraint is required. Thus, applying the uniqueness constraint in this example defines the equivalent of a 1:N relationship between Employee and Department. 1:1 relationships are specified by applying a uniqueness constraint over each role of a binary predicate to ensure that each instance in each column of the corresponding fact table is unique.

ORM allows uniqueness constraints to be verbalised, thus allowing domain experts to participate in their validation. Using the above example, the uniqueness constraint over the '*is employed by*' role box is verbalised as follows.

'Each Employee is employed by at most one Department.'

Visio Modeller, ORM's automated tool support, automatically generates such verbalisations based on the data provided within the fact table. However, the analyst must supply a significant population of instances in order for the tool to determine which column(s) require a uniqueness constraint. In practice, analysts may also apply their knowledge of the domain under investigation when specifying these constraints, rather than relying wholly on data placed in each fact table.

In the example shown in Figure 5.13, the uniqueness constraint spans a single role box, but a uniqueness constraint may spans all roles in a Fact as shown below.

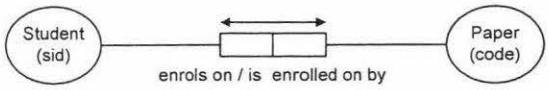


Figure 5.14 – Full Span Uniqueness Constraint

By examination of the associated fact table (shown in Table 5.4), it can be seen that the combination of instances of Student and Paper from each row are never repeated, but taken individually, instances of both Student and Paper are repeated with their respective columns. This indicates that a Student may enrol on many Papers and each

	Student	Paper
1	100	157100
2	100	157123
3	200	157123

Table 5.4 – Fact Table Corresponding to a Full Span Uniqueness Constraint

Paper may be enrolled on by many Students. Thus, the only uniqueness constraint that applies to all possible populations in this fact table is the combination of Student and Paper. As such, this fact population is an example of a M:N relation.

5.7.4 Objectified Associations

When a uniqueness constraint spans all roles as shown in Figure 5.14, it is possible to objectify the association. This means that the association between objects can itself be treated as an object. For example, in Figure 5.15 we could consider the enrolment association between Student and Paper as an object that may be associated with other objects, as shown below.

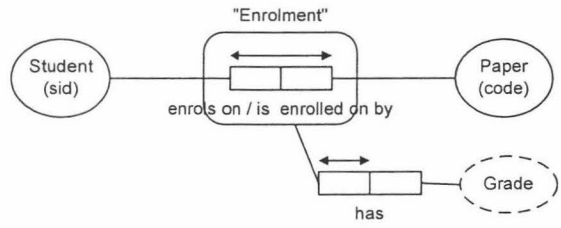


Figure 5.15 – Objectified Associations

The objectified association is indicated by the rectangle that surrounds the predicate being objectified. When mapped onto a relational schema, Enrolment will become a table definition with Grade as its attribute.

5.7.5 Complex Uniqueness Constraints

Combinations of uniqueness constraints are often applied to fact types of arity greater than two. Using the previous example of a quaternary fact type, the following uniqueness constraints would be applied based on the population of the fact table provided.

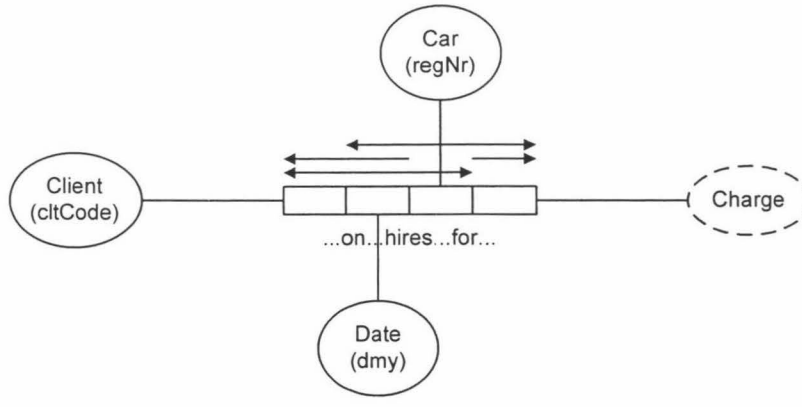


Figure 5.16 – Uniqueness Constraints in a Quaternary Fact Type

Table 5.5 – Fact Table of the Quaternary Constraints

	Client	Date	Car	Charge
1	100	5/12/03	XYZ 123	$100
2	100	5/12/03	ABC 456	$200
3	200	6/12/03	ABC 456	$200
4	300	7/12/03	DEF 789	$150
5	300	8/12/03	DEF 789	$150

The above uniqueness constraints would be verbalised as follows:

1. Given any Client, Date and Car that Client on that Date hires that Car for at most one Charge.
2. Given any Client, Date and Charge that Client on that Date hires at most one Car at that Charge.
3. Given any Date, Car and Charge at most one Client on that Date hires that Car for that Charge.

The first verbalisation corresponds to the uniqueness constraint that spans the first three role boxes (from the left) in figure 5.16. This indicates that the combination of instances associated with the 'Client', 'Date' and 'Car' columns of the fact table (Table 5.5) is unique, but instances of 'Charge' are not. Simple inspection of the fact table proves the validity of this assertion and a similar exercise could be performed on the other two uniqueness constraints.

Analysts must be careful when supplying examples to ensure the data supplied truly reflects the business rules of the domain under investigation. For example, based on the data within Table 5.5, the second verbalisation indicates that a client cannot hire two or more cars at the same charge on the same day. This is probably an invalid assertion for most car hire businesses and highlights the need to involve domain experts to determine the validity of the examples provided.

5.7.6 Optionality

Mandatory Role Constraints indicate which object must participate in a role with another object and is equivalent to the concept of optionality using standard ER terminology.

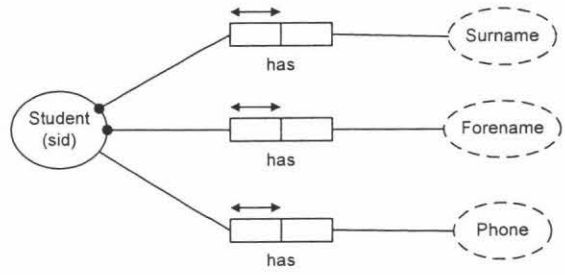


Figure 5.17 – Mandatory Role Constraints

In the example shown in figure 5.17, the object 'Student' must participate in the role with the objects 'Surname' and 'Forename'. This fact is indicated by the dot placed on the line that connects each of these roles to the object. The fact that not all students possess a phone is indicated by the absence of a dot in the student's role with 'Phone' and thus, this role is optional.

In the above example, mandatory role constraints have been explicitly defined for the roles 'Student has Forename' and 'Student has Surname', as indicated by the dots annotating the model. However, all objects must play at least one role in an ORM model. Thus, if an object plays only one role in a model, that role must be mandatory. As such, the objects Surname, Forename and Phone have an implied mandatory role constraint with Student, if it is assumed that they play no other roles. In these circumstances, it is usual practice to omit the mandatory role dot.

As with all other types of constraints, ORM allows the optionality of roles between objects to be verbalised. Using the example depicted in figure 5.17, the following verbalisations describe the optionality of the roles played by a Student with the other objects in this model.

Each Student has some Surname
Each Student has some Forename

Note that no verbalisation is generated for the association between Student and Phone since no mandatory role constraint exists between these two objects.

5.7.7 Set Comparison Constraints

The final class of Integrity Constraint discussed within the Action Assertion type business rule taxonomy, concerns constraints that restrict the way the population of one role relates to the population of another. Such constraints may be classified as Integrity Constraints since they are assertions that must always be true and thus, prohibit actions that violate the specified end-state (GUIDE Business Rules Project 1997, p.27). ORM provides analysts with a rich set of constructs to model these details conceptually, unrivalled by most other modelling approach with the possible exception of UML. However, ORM allows these constraints to be verbalised in natural language fostering end-user participation in the validation of these details. Three kinds of set-comparison constraints are discussed in this section; Subset, Equality and Exclusion.

5.7.8 Subset Constraints

Within ORM, subset constraints are used to enforce that the population of one role must be a subset of another. The example shown in figure 5.18 describes such a

situation in which any employee who is allocated a computer must also be an employee who has been allocated an office. The assumption in this example is that the computer in question is a PC (rather than laptop) and thus needs a permanent location.

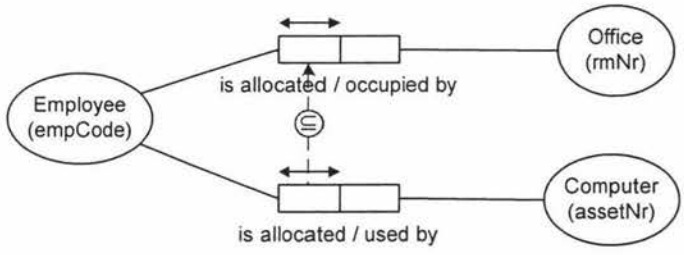


Figure 5.18 – Subset Constraint

The subset constraint is represented by a dashed arrow that runs from the subset role to the superset, annotated with a subset symbol ⊆. The constraint is applied against the populations associated with the roles:

Employee is allocated Office, and
Employee is allocated Computer

By examination of the corresponding fact tables for these two roles (shown in Table 5.6), it can be seen that their populations mirror this subset constraint. Each instance of Employee in the fact table illustrating the fact 'Employee is allocated Computer' is also an instance of Employee in the fact table illustrating the fact 'Employee is allocated Office'. However, an Employee who has been allocated an Office may not always be allocated Computer.

Table 5.6 - Fact Tables Corresponding to Subset Constraint

Employee is allocated Office

	Employee	Office
1	100	5E20
2	200	5E23
3	300	5C22

Employee is allocated Computer

	Employee	Computer
1	100	IT 02345
2	200	IT 23492

Having defined the subset constraint illustrated in Figure 5.18, ORM would verbalise the constraint as follows:

'If Employee e is allocated some Computer then Employee e is allocated some Office. '

Note that ORM employs an If ... Then construct to express subset constraints, indicating the conditional nature under which such constraints are applied. Conditions, as they relate to the BRG's classification scheme for Action Assertion are discussed in section 5.7.17, when the role of If .... Then constructs will be discussed in more detail. The above verbalisation also employs simple algebra in that an instance of Employee is referred to as 'e'. ORM typically uses the first character of the object whose population is constrained to articulate these algebraic expressions. However, when validating these constraints with end-users in the case study for this thesis, it was found that by replacing such symbols with an actual value from the Fact table improved the comprehensibility of the verbalisation.

5.7.9 Equality Constraints

An equality constraint asserts that the population of one role must equal the population of another. The example shown in figure 5.19 describes a scenario in

which the set of employees receiving a bonus equals the set of employees who have exceeded a sales target.

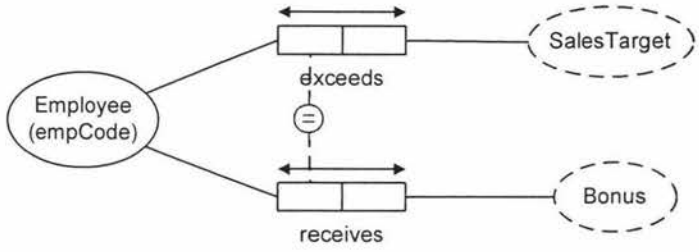


Figure 5.19 – Equality Constraint

Equality constraints are only applicable when the roles being compared are optional, as shown in figure 5.19. If both roles are mandatory, an equality constraint is implied as illustrated in figure 5.20.

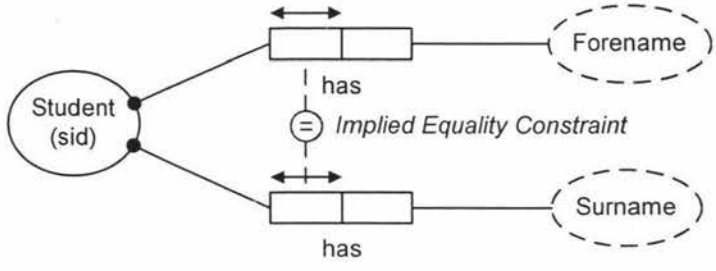


Figure 5.20 – Implied Equality Constraint

Since each role is mandatory (as indicated by the dots linking the roles to the Student object), all students who have a forename must also have a surname and the converse must equally be true. In such situations, an equality constraint is not usually defined explicitly as shown in the example. Note also that if one role is mandatory and the other is optional, an equality constraint cannot exist between them since it is possible that not all instances of the object play the optional role.

Using the example in figure 5.19, the equality constraint would be automatically verbalised by Visio Modeller as the following.

'Employee e receives some Bonus if and only if Employee e exceeds some Sales Target.'

Again, it was found by replacing the algebraic symbol, 'e' in this example, with an actual instance of the object in question, promoted greater understanding of the assertion by domain experts.

5.7.10 Exclusion Constraints

The final category of set comparison constraints that may be modelling using ORM constructs is the exclusion constraint. This constraint asserts that the populations of two optional roles played by the same object are mutually exclusive, as shown below in figure 5.21. The example illustrates a situation where a customer may be awarded air points or a gift, but not both.

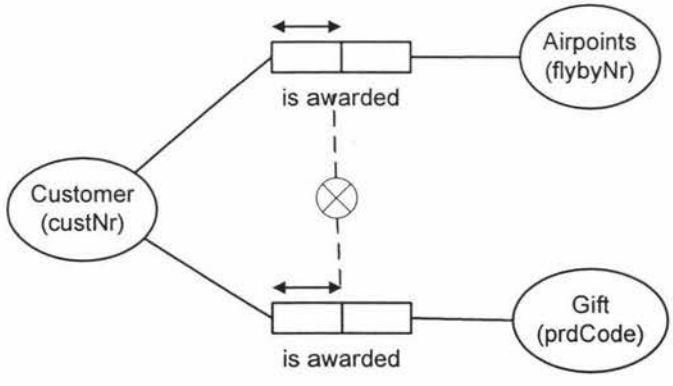


Figure 5.21 – Exclusion Constraint

ORM's verbalisation of the above constraint would be as follows.

'No Customer that is awarded some Airpoints is awarded some Gift.'

Occasionally, a disjunctive mandatory role constraint is combined with exclusion constraints to form an exclusive-or constraint. When applied together in this way, such a constraint asserts that an object *must* participate in one of the two roles between which the constraint applies. The example depicted in figure 5.22 illustrates an exclusive-or constraint in which a payment must be either in the form of a cheque or credit card, but not both.

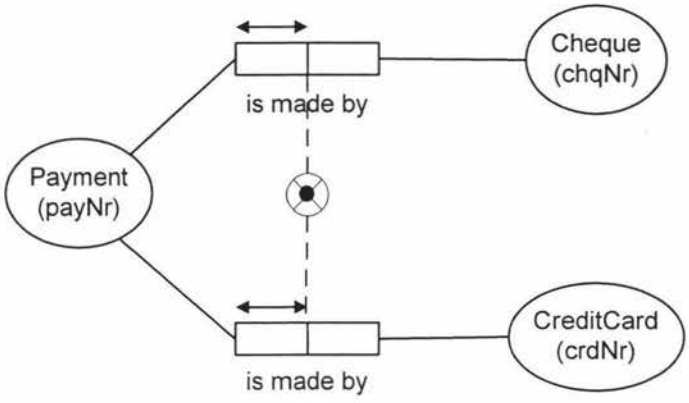


Figure 5.22 – Exclusive-or Constraint

ORM verbalises the above constraint as two separate constraints; one verbalisation for the exclusion constraint and another for the disjunctive mandatory constraint. Using the example in figure 5.22, the following verbalisations would be generated.

> 'No Payment that is made by some Cheque is made by some CreditCard.'
> 'Each Payment is made by some Cheque or is made by some CreditCard.'

The first verbalisation asserts the exclusion constraint and the second asserts the disjunctive mandatory role constraint that exists between these two roles.

5.7.11 Join Constraints

In this final section on set comparison constraints using ORM constructs, the ability to define conceptual joins across compatible role-sequences is discussed. Very few data modelling tools or techniques provide analysts with such flexibility, although in the next section, UML's constructs in this regard will be compared to that of ORM's.

Conceptual joins are formed by defining compound fact types between which a constraint is applied. In the example illustrated in figure 5.23, the shaded role-boxes

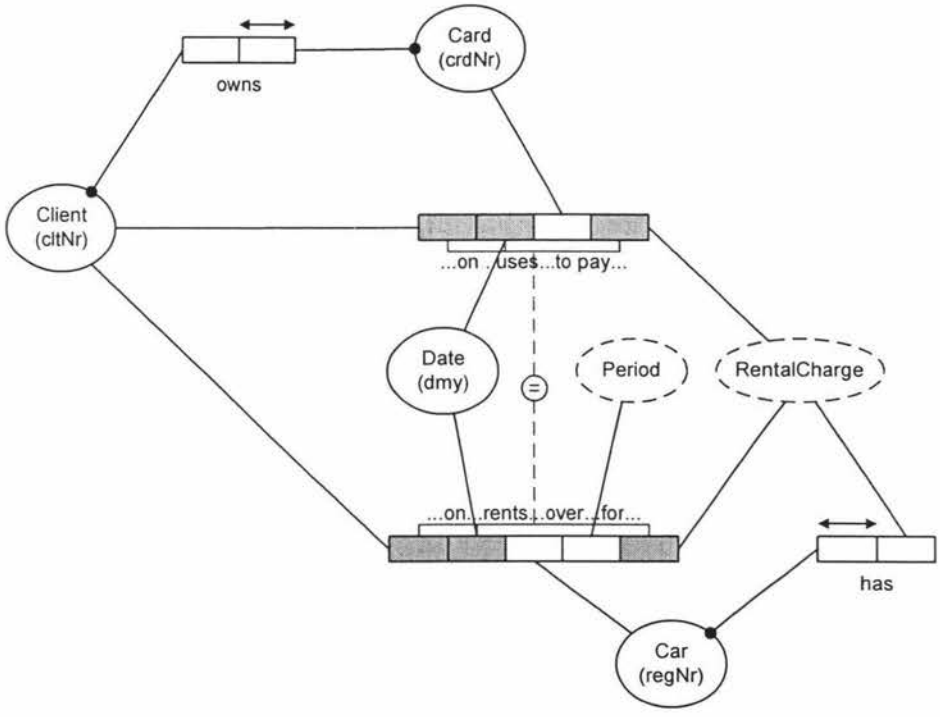


Figure 5.23 – Join Constraint

indicate which roles participate in the conceptual join. The join path is indicated by the solid line that connects these shaded role-boxes. Thus, in the predicate '...on...uses...to pay...', the role-boxes associated with instances of 'Client', 'Date' and 'RentalCharge' have been conceptually joined. Similarly, in the predicate

'...on...rents....over....for...', the role-boxes associated with 'Client', 'Date' and 'RentalCharge' have also been joined. Between these two conceptual joins, an equality constraint has been applied that asserts that the populations over these join paths are equal. Verbalising the above constraint produces the following assertion.

> Client c on Date d rents some Car over some Period for RentalCharge r if and only if Client c on Date d uses some Card to pay for RentalCharge r.

Thus, the values of the instances associated with 'Client', 'Date' and 'RentalCharge' must be the same in both the corresponding fact tables of this compounded fact. To improve the clarity of assertions of this complexity for domain experts, it is again suggested that actual instances from the corresponding fact tables are used instead of the symbols 'c', 'd' and 'r'.

5.7.12 Set Comparison Constraints in UML

UML is one of the few modelling approaches that allow analysts to represent set comparisons between populations. However, it will be demonstrated that its ability to apply such constraints is limited by comparison to the constructs provided within ORM. Some constraint details cannot be modelled at all in UML and others can only be captured as textual annotations to graphical models.

5.7.13 Subset Constraints

ORM allows subset constraints to be modelled graphically between any pair of compatible roles sequences and provides a corresponding verbalisation of the constraint. The example shown in figure 5.18 demonstrated the graphical notation applied to represent this constraint. By comparison, UML allows subset constraints to

be specified only for whole populations and provides no graphical notation for this constraint when applied between single roles or between parts of an association (Rumbaugh et al., 1999). When specifying a subset constraint between whole populations in UML, a textual annotation must be provided by the analyst, as illustrated in figure 5.24. The example shown below corresponds to the ORM representation of the same scenario in figure 5.18.

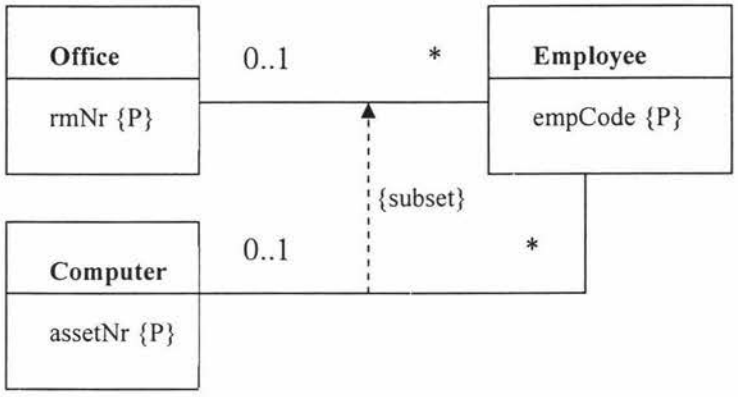


Figure 5.24 – UML and Subset Constraints

5.7.14 Equality Constraints

Within ORM, equality constraints are the logical equivalent of two subset constraints (one in each direction) between role sequences. Since UML has no graphical notation for this constraint (Rumbaugh et al., 1999), it is possible that analysts could use this approach to capture such details. However, such a solution would be 'messy' and could only be applied in those situations where the constraint exists between whole populations.

5.7.15 Exclusion Constraints

UML does not provide graphical notation for exclusion constraints (Rumbaugh et al., 1999), but does allow analysts to capture exclusive-or constraints to model those situations where each instance of a class plays exactly one association role from a set of alternatives. Such situations are modelled by using the textual annotation '{xor}' adjacent a dashed line that connects the relevant associations. Using the ORM example depicted in figure 5.22, the equivalent UML model has been illustrated in figure 5.25.

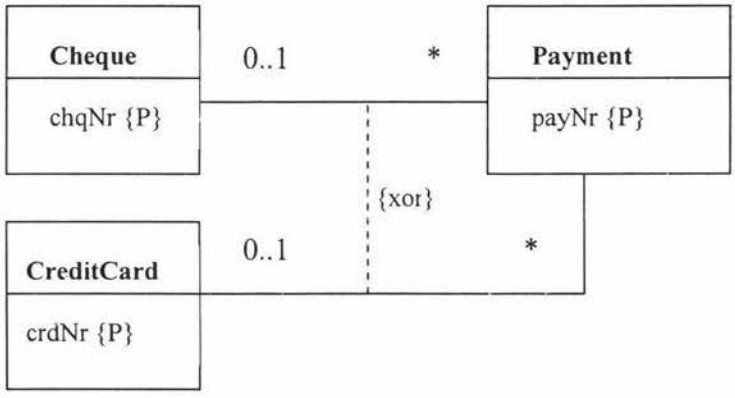


Figure 5.25 – UML and XOR Constraints

5.7.16 Join Constraints

Although the need to specify join constraints frequently arise in applications, very few modelling tools or techniques possess a graphical notation to represent them, including UML. If an attempt was made to model the example shown in figure 5.23 as a UML representation, the analyst would only be able to provide a textual annotation in the form of a comment attached to the relevant model elements. This textual annotation could include a comment that verbalises the constraint in a similar form to the verbalisation automatically generated by Visio Modeller. However, the

analyst must provide such details themselves as well as the code necessary to implement the constraint (Rumbaugh et al, 1999). By comparison, ORM allows join constraints to be captured easily, providing not only a graphical notation for such constraints but will also verbalise these details in a form that facilitates validation by domain experts. Once validated, these constraints can then be mapped automatically by Visio Modeller into a logical schema. From the logical schema, Visio Modeller will allow analysts to generate the database and its constraints.

5.7.17 Conditions

As previously described in section 5.71, conditions are Action Assertions that assert that if something is true, another business rule will apply (GUIDE Business Rules Project 1997, p.27). Often, these assertions will be expressed in the form 'If .....Then....' Although ORM employs the 'If...Then...' construct to specify subset constraints and the 'if and only if' construct to specify equality constraints, these fall into the Action Assertion category of integrity constraints, rather than the condition category. What distinguishes integrity constraints from conditions is that the former have immediate enforcement power and prohibit actions that would cause a false truth value, whereas conditions are tests performed against values. Two condition type Action Assertions examples are provided below.

> 'If a rental car's mileage exceeds 100,000 km then that car must be replaced.'
> 'If a driver's age is below 25 then charge the highest insurance premium.'

Constraints in this form can often be at least inferred using ORM's subtyping constructs in conjunction with value constraints. Value constraints are applied against objects and define discrete values, or a range of values, that instances of that object

may hold. The example shown in figure 5.26 demonstrates how a value constraint may be used. The value constraint in this example is used to ensure that the sex of student is defined as either 'M' or 'F'. However, this constraint alone does not prevent instances of the 'Male' subtype from having a Sex type of 'F'. To prevent such

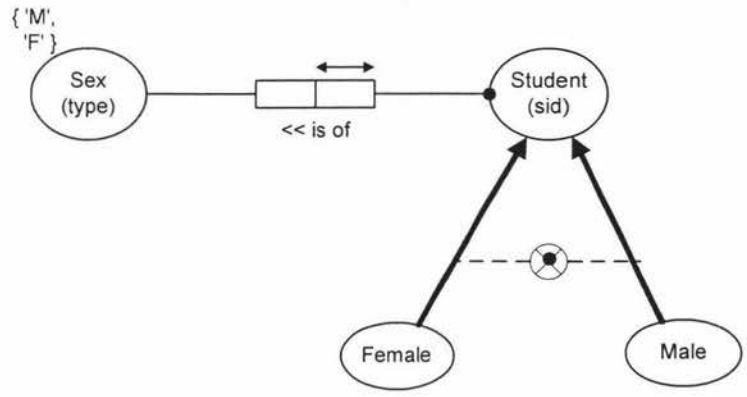


Figure 5.26 – Applying Value Constraints to Specify Conditions

occurrences, analysts must provide a subtype definition that specifies at least one role played by its supertype(s). Such definitions are formal rather than just simply comments. Analysts may use the operators 'is a' or 'is an' to mean 'is defined as', and 'who', 'that', or 'which' after the subtype name. Optionally, 'each' may be used before the name of the subtype being defined. Using the example in figure 5.26, the following subtype definitions could be provided for Male and Female, respectively.

Each Male is a Student who is of Sex 'M'
Each Female is a Student who is of Sex 'F'

Thus, it may be inferred from the above definitions that if a Student is Female, then their Sex type must be 'F'. Therefore, the combination of a value constraint and subtype definitions can allow certain forms of conditions to be at least inferred.
In other situations, conditions may need to be specified as a textual rule in the same manner as textual annotations are used to express certain rules in UML. ORM simply provides guidelines as to how such rules are expressed by suggesting that a high-level formal language be employed. The elements within an ORM model affected by a textual rule are indicated with a circled 'R', often numbered, to signify that extra rules apply. The example in figure 5.27 illustrates how a textual rule can be applied to

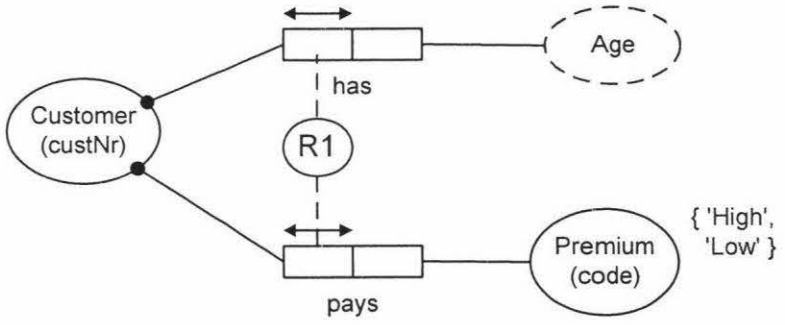


Figure 5.27 – Applying Textual Rules to Specify Conditions

indicate a *restricted* equality constraint. This rule could be defined is as follows.

R1 If Customer has Age < 25 then Customer pays Premium 'High'

Thus, this constraint specifies that the population of instances of Customer who have a Premium code of 'High' is equal to the population of instances of Customer whose Age is < 25.

Although textual rules provide a useful mechanism for expressing those constraints for which no graphical symbol exists, ORM does not provide automated tool support

for mapping these constraints into either logical or internal schemas. Further, since it is left to analysts to decide how such rules are expressed, there is no guarantee as to the rigour in which these rules are articulated or how comprehensible they will be for domain experts. However, if analysts adopt the guidelines for expressing textual rules in a formal manner using natural language, applying textual rules should facilitate the capture of most condition type Action Assertions at the conceptual level.

5.7.18 Conditions in UML

UML provides no constructs that can be applied to express conditions other than as textual annotations to the model (Rumbaugh et al, 1999). In this sense, the approach taken is similar to that of ORM in the case of textual rules. However, there are no guidelines as to how such rules are expressed or how they may be graphically represented.

5.7.19 Authorisations

As previously stated in section 5.7.1, authorisations are a category of Action Assertion type business rules. Such rules specify who or what may carry out a specific action and as such can be considered as a declaration of privileges associated with a particular object capable of independent activity (GUIDE Business Rules Project 1997, p.27). The BRG have suggested that such rules be expressed in the form:

(Only) X may do Y

Since ORM models domains as objects playing roles, many of the business rules falling into this category could be expressed using basic ORM constructs. Thus, if an object plays a certain role that is not played by any other object, then such facts imply

an authorisation type Action Assertion. For example, if only managers are allowed to hire consultants, this fact could be documented as follows.

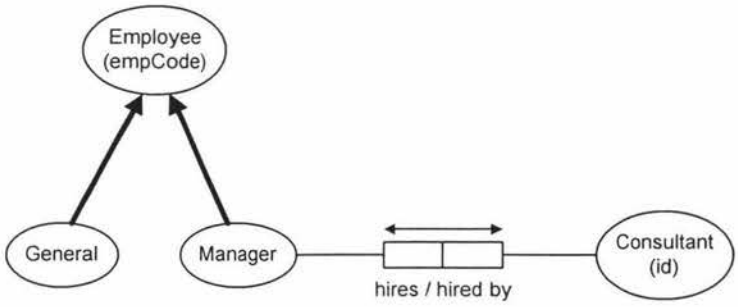


Figure 5.28 – Defining an Authorisation

Assuming that no other object plays this role with Consultant, an authorisation type business rule is implied.

In other situations where only specific instances of an object are authorised to carry out some action, the approach described in section 5.7.17 could possibly be applied. Thus, by combining value type constraints with formal subtype definitions, it is possible that authorisation type rules in this form could be expressed. Finally, analysts also have the option of applying a textual rule, also described section 5.7.17. Textual rules could be applied in more complex situations for which ORM provides no graphical symbol to represent the desired constraint.

5.7.20 Authorisations in UML

Analysts employing UML to express authorisation type Action Assertions would need to rely mostly on textual annotations to document these details. However, like ORM, some forms of authorisation could be implied simply from the associations

between classes. Where an authorisation rule is applicable only for specific instances, a textual annotation would always be required.

5.7.21 Conclusions Regarding ORM and Action Assertions

Action Assertions include categories of business rules that would be difficult or even impossible to define using traditional data modelling tools and techniques. In particular, set comparisons (a form of integrity constraint), conditions and authorisations would challenge the descriptive powers of most data modelling approaches. By contrast, ORM possesses a rich set of constructs for analysts to employ. Certainly, rules falling into the integrity constraint category of Action Assertion can be captured with little difficulty using ORM constructs. For example, the ability to define conceptual joins facilitates the expression of complex, but still quite common forms of integrity constraints. In stating this, it is recognised that the condition and authorisation categories of Action Assertion do present challenges, even when using ORM. The reason for this is that like many other data modelling tools and techniques, ORM employs constructs that allow constraints to be applied to whole populations, rather than specific instances. ORM is more flexible than most other approaches in this respect in that constraints can easily be applied against populations associated with *each individual role* that an object plays. This is made possible because ORM makes no initial distinction between entities and attributes, promoting a conceptually simpler model. However, condition and authorisation type Action Assertions are often associated with the testing of values of specific instances from a population. Although ORM appears capable of expressing these forms of constraints, many rules falling into these categories could only be captured as a

textual rule annotating the general model. As such many of these rules would require manual implementation.

Although all data modelling approaches allow optionality and cardinality details to be captured, very few possess the linguistic properties that allow these details to be expressed naturally. For example, Barker's notation and ORM are two of only a few data modelling approaches that facilitate inverse reading of associations between objects/entities. This problem becomes even more apparent when other, more complex constraints such as set comparisons are modelled (ter Hofstede et al, 1994). When applying ORM however, all constraints with the exception of those that must be modelled as textual rules are automatically verbalised in the natural language of FORML (Formal Object Role Modelling Language). Occasionally, these verbalisations may require some modification in order to facilitate greater understanding amongst the end-user community. This point was illustrated in the discussion regarding conceptual joins where Visio Modeller, ORM's automated tool support, employs algebraic symbols to refer to specific instances of an object. However, these symbols can easily be replaced by substituting actual values from the fact tables associated with the conceptual join. This would allow even quite complex constraints to be validated by domain experts that should promote the quality of the resulting data model.

5.8 DERIVATIONS

The final category of business rule defined by the Business Rules Group (BRG) concerns Derived Facts; facts that can be derived or logically inferred from Terms, Facts, other Derivations or Action Assertions (GUIDE Business Rules Project 1997,

p.31). A Derivation is the actual mechanism by which a Derived Fact is either mathematically calculated or logically inferred. Figure 5.29 defines the structure of Derived Facts and Derivations, as proposed by the BRG. It can be observed from the model that both Derived Facts and Base Facts are a type of Fact. Base Facts are simply assertions concerning the business domain that are stored by the system and have been previously discussed in section 5.6.2

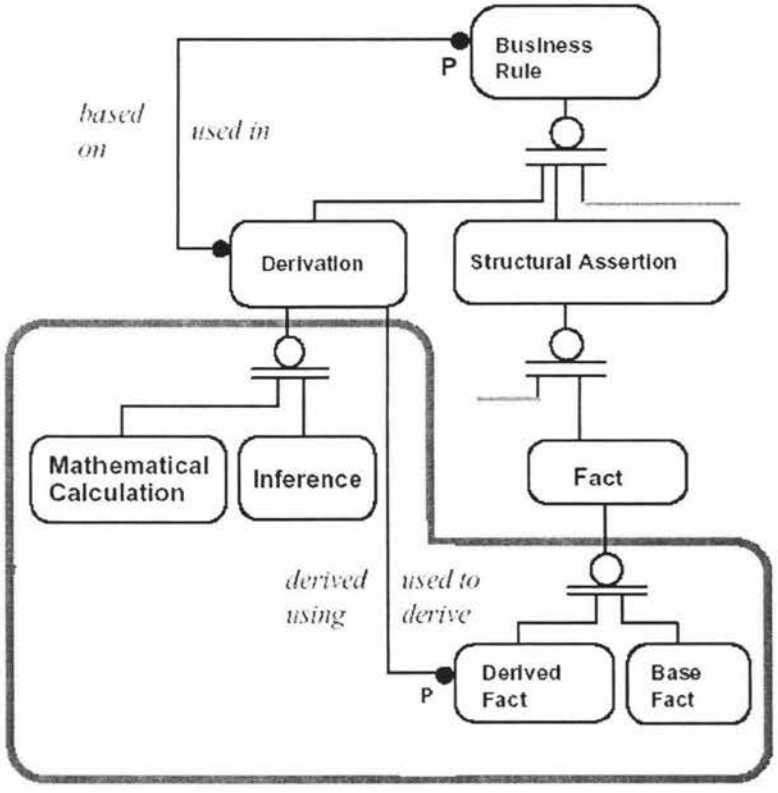


Figure 5.29 – The Composition of Derived Facts and Derivations[5]

Derived Facts are usually not stored within an organisation's database unless there are compelling reasons to do so; for example, to improve the performance of a system due to the complexity of the Derivation algorithm. In such situations, it may be more desirable to permanently store the results of the Derivation rather than to re-calculate

[5] Reproduced from GUIDE – Business Rules Project, Final Report, 1997, p.31

those results each time the end-user requires them. However, irrespective or whether Derived Facts are permanently stored, details of the Derivation need to be captured by the analyst and validated by domain experts. Unfortunately, Derivations are another category of business rule that cannot be represented in most traditional ER modelling tools, although such details often appear within process descriptions of Data Flow Diagrams(Gane & Sarson, 1979) and are typically defined using Structured English. Although Structured English may be appropriate as a means of communication between analysts, more complex Derivations may incorporate constructs (e.g. loops) that may be unfamiliar to domain experts.

By contrast, both ORM and UML allow derived elements to be captured within the data model. Figure 5.30 details an invoicing system using ORM constructs in which the objectified association, InvoiceItem, is associated with a Derived Fact, 'InvoiceItem has Subtotal' and Invoice is associated with the Derived Fact, 'Invoice has Total'. Since these Derived Facts are unlikely to be permanently recorded in the eventual database, they have not been displayed within the model. However, the mechanism by which these Derived Facts are determined, are identified, as shown below.

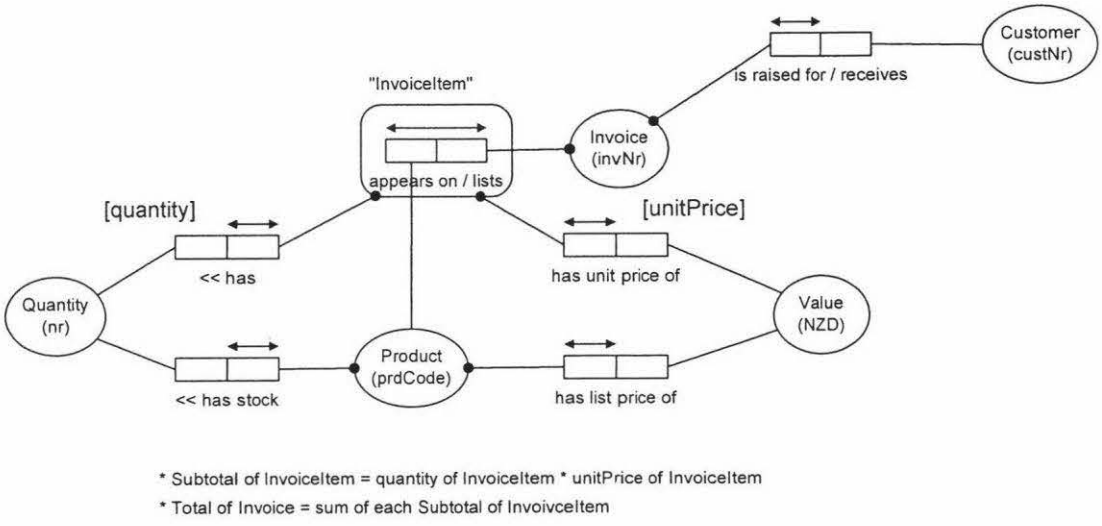


Figure 5.30 – Derivations within ORM

Each derivation is identified by an asterisk followed by the method of calculation and incorporates both the objects and roles associated with the Derived Fact. ORM allows analysts to define *rolenames* in addition to the required predicate name. This can improve the readability of attribute names that are automatically generated during the mapping to attribute-based models or relational schemas. Rolenames may also used as attribute names when defining Derivations, appearing on the diagram within square brackets next to their respective roles. The rolenames 'unitPrice' and 'quantity' have been applied in this manner, as shown in figure 5.30 and also appear in the Derivation itself in order to calculate the invoice item subtotal and invoice total, respectively. The Derivation itself is articulated in ORM's formal language and in a previous version of ORM's automated tool called InfoModeller, these Derivations could then be mapped automatically into the equivalent SQL statements (Asymetrix Corporation, 1994). Unfortunately, the current version of this tool, Visio Modeller, does not support this translation and so this task must be performed by the developer. But perhaps more importantly, from the perspective of validation, these Derivations are

articulated in a form that end-users should be able to comprehend. However, one could imagine the predicate-based format of Derivations within ORM may appear to be rather verbose in more complex calculations. In such circumstances, a more informal expression of these rules, based on only defined rolenames, may be a more suitable format for domain experts (Halpin, 2001). Appling this strategy using one of Derivations in the example given in figure 5.30, the following Derivation may be defined.

SubTotal = Quantity * UnitPrice

5.8.1 Derivations in UML

In UML, derived elements are indicated by prefixing the name of the element with a forward-slash '/'. The derivation rule itself may also be indicated as a textual annotation connected to the derived element by a dashed line (Booch; Rambaugh, and Jacobson, 1999). Figure 5.31 illustrates how the InvoiceItem subtotal, used in the previous example in figure 5.30, would be represented using these UML constructs.

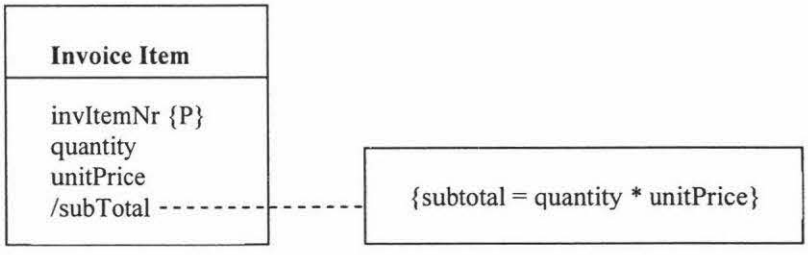


Figure 5.31 – Derivations in UML

Since UML employs the use of attributes and association role names, Derivations are typically specified more concisely than is usually possible in ORM's predicate-based

format. However, ORM also allows rolenames to be used as attribute names, as previously stated, resulting in Derivations being expressed a similar format to the UML representation. The main disadvantage of employing the attribute-based format is decreased stability, since attributes may be remodelled as associations by the analyst as the data model is successively refined. The predicate-based notation does not suffer from this lack of stability. Thus, the analyst must make a decision involving a trade-off between the conciseness of the attribute-based notation and the stability of the predicated-based notation.

5.9 CONCLUSION

The primary purpose of this chapter has been to examine of the central concepts concerning ORM and Business Rules Modelling (BRM), with the aim of synthesizing these two approaches into a single conceptual framework. The motivation to attempt this synthesis was to provide a means of expressing business rules formally, but still comprehensible to domain experts. If ORM constructs can be adopted to provide a formal natural language to express business rules, then this suggests that domain experts would be more able to participate in their validation, thus promoting the quality of the resulting system requirements. Although business rules do not present any new ideas that are not already familiar to analysts, it was suggested that they do provide a useful framework to discuss the system with domain experts. Business rules originate as policy statements issued by management and as such, domain experts are as familiar with them as analysts who attempt to model them. It is suggested that this fact alone provides a strong incentive to model the domain within a business rules framework.

Initially, this chapter identified and defined the key concepts associated with Information Systems Development (ISD). These definitions were then applied to both ORM and BRM in order to place them in the context of ISD. As a consequence, it was suggested that ORM is a prescriptive data modelling technique for defining a conceptual schema during the requirement analysis stage of ISD. In contrast, BRM was defined as a descriptive data modelling tool for describing the structure and behaviour of organisations.

The remainder of this chapter then focused on a detailed description of the business rules taxonomy provided by the Business Rules Group (BRG). The group has suggested that business rules may be categorised into definitions of business terms, Structural Assertions, Action Assertions and Derivations. Each business rule category was defined and it was demonstrated how ORM constructs may be applied to express rules from each of these categories. It was explained that although traditional data modelling tools are able to represent the basic structural details of an organisations, as well as capturing the cardinality and optionality of associations between these structural elements, other forms of business rules could not be captured. This was particularly the case for the Action Assertion and Derivation categories of business rules. The fundamental problem with traditional data modelling tools in this respect is that they simply do not possess the necessary constructs with which to express those categories of business rule. For example, set comparisons, overlapping subtypes, conditions and join constraints, all require constructs that most tools and techniques do not possess(ter Hofstede et al, 1994).

By comparison, both ORM and UML possess a rich set of constructs that allow analysts to capture most categories of business rules. However, of these two techniques, only ORM possesses a linguistic framework that allows business rules to be expressed in a formal natural language. Although UML will allow analysts to annotate their models using textual notes, only ORM is able to generate most of these verbalisations automatically. This ability ensures the completeness of the business rule as well as providing the basis on which to map these rules to the target DBMS. The verbalisation of business rules should also facilitate end-user validation by improving the transparency of the requirements analysis activity.

CHAPTER 6

ANALYSIS OF FINDINGS

6.1 INTRODUCTION

The purpose of this chapter is to present the findings of the research undertaken during the course of this thesis. It will discuss what was learnt about Object Role Modelling (ORM) and the modelling of business rules during the application of these techniques within a Private Training Enterprise (PTE). This chapter also examines the problems encountered during the case study and action research components of this thesis and the attempts resolve these difficulties. Finally, the chapter concludes with an assessment as to the value of employing ORM as the language to express business rules to domain experts.

6.2 SUMARY OF RESEARCH UNDERTAKEN

Chapter 4 argued that action research, embedded within an explanatory case study, provides a suitable framework to explore the validity of the hypothesis of this thesis. This hypothesis is that Object Role Modelling constructs can be used to articulate business rules in a format that facilitates their validation by the end-user community. The process began with the first case study that sought to determine the business rules that pertain to a small sub-system of a Private Training Enterprise (PTE). This activity was then followed by an action research component that involved a collaborative

effort with end-users and management to validate these business rules and transform them into the formal natural language associated with ORM. Completing this step provided the system requirements for the PTE's sub-system. Using these requirements, the sub-system was developed into a final product that was then evaluated by end-users and management in the second case study. Based on the observations and comments drawn from this case study, the effectiveness of employing ORM to articulate business rules was assessed. The findings determined in each of these research activities are now discussed in detail beginning with an analysis of the first case study.

6.3 THE FINDINGS OF THE FIRST CASE STUDY

As previously stated above, the purpose of the first case study was to discover the target organisation's business rules in relation to a new sub-system. The aim of the sub-system was to provide supervisors with a means to mentor students during the course of their studies. Further details on the nature of this sub-system can be found in section 4.2.4.3.

6.3.1 The Case Study's Initial Focus

The initial focus of the case study was to place the mentoring sub-system into context with the organisation's other systems. This was felt necessary since the preliminary interviews with management during the project selection indicated that the mentoring sub-system would require student data from many of the PTE's existing sub-systems e.g. Student Enrolment, Attendance Monitoring and Results Processing. Semi-

structured interviews were used as the primary source of data during this initial activity, the details from which were used to construct the context level DFD diagram shown in Appendix 1. This diagram was then validated by end-users before moving to the lower level processing details of each sub-system.

6.3.2 Determining Lower Level Processing Details

As documented in section 4.3.6, it was felt that DFD's would not be appropriate as a tool with which to validate these lower level details with end-users. This of course is also true of context diagrams, but due to their high level focus and their low level use of symbolic notation, the concepts they contained were relatively straight-forward to convey to both management and end-users. To document lower level sub-system processing details, Oracle's Business Process Modelling (BPM) tool was employed. BPM has been used by business analysts for a number of years and pre-dates their automated tool support that is now the norm. Although end-users were unfamiliar with this modelling tool, most were able to appreciate the content of these models with only a 15 minute explanation. The most common comment with regard to these models was that, "*They look like a flow chart turned on its side.*" Once end-users appreciated this fact, it required little effort to explain the additional concepts of 'swim lanes' and triggers. This modelling tool proved to be a useful mechanism for summarising the details drawn from the semi-structured interviews and for identifying some of the terminology employed by end-users. These business process models are shown in Appendix 2. Management also appreciated the efficacy of this modelling tool in terms of documenting all of the organisation's processes. As such, they have adopted business process modelling as an organisational standard for documenting processes and are now able to construct their own models within MS Word. This was

an unanticipated benefit for management since prior to the commencement of this research, very little documentation of their activities existed, other than that contained within procedure manuals that were often out of date.

6.3.3 Identifying Business Rules Using BPM

The construction of the BPM's initiated the collection of business rules. As briefly described above, the models help to identify some of the terminology used within the PTE. Terms, as explained in section 5.4.1 are a category of business rule and are used in the expression of all other forms of rule. As such, it was considered crucial that not only were these terms identified and recorded, but their precise meaning in the context of this organisation be defined. As such, it was considered appropriate to incorporate the technique of Grounded Theory by allowing these semantic details to emerge from multiple sources of data through iterative analysis. These sources included domain experts, internal documents, WordNet and the business process models that had initially identified some of these terms. However, it was the domain experts who ultimately decided upon the precise meaning that they would apply to each of these terms. The role of the researcher, at least in this context, was simply to provide guidance to the domain experts in reaching a consensus. Once the shared meanings of the terms had been established, their definition was recorded in Visio Modeller using the general form <Term> is a... However, it was evident after this exercise had been completed that most of the terms identified related to system-wide concepts, such as Student, Enrolment and Programme. It was clear that many other terms had not been identified from the examination of the existing system and would not be until the specific requirements of the new mentoring sub-system had been determined. This does not mean that the process modelling exercise was not without value as the

benefits identified in the list below indicates.

- Identified the data that the current system records,
- Identified some of the terminology employed in the organisation
- Identified the probable interfaces between the new sub-system and the current
- Facilitated the familiarisation with the organisation's activities
- Identified the problems associated with the current system
- Initiated a working relationship with the domain experts.
- Assisted in the identification of business rules associated with each task carried out to perform specific processes (as documented in Appendix 2).

6.3.4 Finding Examples

(Halpin, 2001) describes the process of conceptual data modelling as beginning with the transformation of *familiar* examples into elementary facts. Examples are found by the examination of forms, reports and other documents within the problem domain. However, this could be problematic in some new systems since such documents may not exist. This theme is discussed more fully in section 6.4.1. Initially, the lack of familiar examples was perceived as a potential problem within the first case study after the construction of the BPM's. Although a mentoring procedure was already in place (as detailed in Appendix 2), supervisors lacked readily available information to guide students working towards the completion of a qualification. The aim of the new sub-system was to provide supervisors with this missing information. However, the question arose: where would the examples come from to initiate the process of documenting the business rules relating to the new mentoring sub-system? Fortunately, further analysis revealed that the supervisors would often draw upon their

existing knowledge of the National Qualification Framework (NQF) to guide students as to the appropriate options in terms of the Unit Standards that could be used to meet the specific requirements of a qualification. However, even staff familiar with the framework would often need to refer to a number of New Zealand Qualification Authority (NZQA) documents during the mentoring process to find the specific information they required. This information is usually in the form of a MS Word document that explains the specific requirements of a particular qualification. Since each qualification on the NQF is documented in this way, a source of examples was discovered. How this information was used is described in section 6.3.8.

6.3.5 Defining the Output from the New Sub-System

In addition to the documentation available from the NZQA, simple reports could also be generated from an MS Access database system that pre-dates the current system. Some supervisors still used this old system to determine the unit standards achieved by students, as detailed in section 4.2.4.3. Although, the reports generated from this system were helpful in providing supervisors with some of the information they required during the mentoring process, it was felt that the first task should be to re-design these reports to more truly reflect the supervisor's requirements. This conclusion was reached after discussions with the supervisors who believed that a well constructed report would be able to quickly convey the information they needed to guide their students. In this sense, the requirements for the new sub-system were initially based upon what information that sub-system should be able to generate. Knowing the output from this new sub-system thus assisted in determining what business rules would be pertinent in defining its requirements. This approach also helped to establish the parameters of user-acceptance testing. If the new sub-system is

able to generate the reports required by the supervisors, it suggests that this sub-system has met its requirements.

By examination of the reports generated from the old sub-system and using input from the supervisors as to the content required in the reports generated from the new sub-system, a prototype report was defined using MS Word (shown in Appendix 3). It was determined that the new reports would consist of 2 major elements. The head of the report would consist of a summary section that identified each of the qualification requirements and the student's progress in meeting those requirements. This section would then be followed by the second element that would consist of a number of report sections. Each of these sections would examine one of the qualification requirements identified in the summary section by listing all of the unit standard options available to meet that requirement and an indication as to which of these unit standards had already been achieved by the student. Thus, the number of report sections in this second element of the report would be determined by the number of individual requirements associated with the qualification in question.

The aim of summary section at the head of the report was to provide supervisors with a simple overview of the qualification requirements and the student's progress in meeting them. It would identify the credit totals associated with each requirement and state whether or not the student has met each of them and the requirements for the qualification as a whole.

The body of the report aimed to provide supervisors with a detailed break-down of each requirement. Supervisors would use this information to guide students who had

not met specific qualification requirements by indicating to them their options in terms of unit standards that could be used to fulfil those requirements.

6.3.6 Determining the Scope of the Analysis

Constructing the prototype report shown in Appendix 3, helped to further refine what business rules would be pertinent to the data requirements of the new sub-system. These details were augmented with notes taken during the direct observation of the mentoring process, the semi-structured interviews and the business process models constructed to document these activities (as shown in Appendix 2). From these details, it was discovered that the structure of the NQF is based on a 4 tier hierarchy of Fields, Subfields, Domains and Unit Standards. A Field is defined as a broad area of learning that consists of a number of Subfields. Subfields are a narrower area of learning that consists of Domains. Finally, at the lowest level, there are Unit Standards. Qualifications can exist at the Field, Subfield or Domain level and most often specify requirements that involve students achieving Unit Standards from within specified Subfields or Domains. Thus, in order to determine whether a student has met a particular requirement, it is necessary to determine the Subfields and Domains to which their Unit Standard achievements belong. As a consequence of this analysis, it became clear that in order to provide the information the supervisors need to guide their students, the business rules relating to the following would need to be determined.

1. The structure of the National Qualification Framework (NQF)
2. The structure of qualifications and their associated requirements
3. The associations between the NQF and qualification requirements

4. The associations between student achievements and qualification requirements

To determine the business rules relating to the above, it was decided that the information provided by the New Zealand Qualification Authority (NZQA) on their web-site would first be examined. This approach allowed many of the business rules relating to the mentoring sub-system to be captured first and these details were later augmented with additional rules specific to the Private Training Enterprise. The NZQA's web-site proved to be a valuable source of information accounting for approximately 75% of the total business rules captured for the new sub-system as a whole. Not only does the web-site allow users to browse the NQF to determine its structure, but it also provides a detailed glossary of terms associated with that structure as well as the specific requirements associated with each qualification.

6.3.7 Defining Business Terminology

The next task was to document each of these terms in Visio Modeller using the same approach as described in section 6.3.3. These terms were then presented to end-users and management to ensure that each definition was expressed in a form meaningful to both parties. This was considered an important step since these terms were used to express other forms of business rules in the next phase of the analysis effort. The exercise also proved to be useful in other ways since it was clear during this activity that even domain experts often possess misconceptions as to the true meaning of the terminology they employ in their day-to-day activities. Management also saw the value that this exercise afforded in the generation of a glossary of business terminology stating that, "It will be a useful for new staff or staff taking on a new role". The glossary produced from this exercise has been included in Appendix 4.

6.3.8 Expressing Business Rules

The next stage was to incorporate these terms in the expression of the business rules that would define the requirements of the new sub-system. As previously stated in section 6.3.4, a source of examples had been discovered in the form of MS Word documents prepared by the NZQA that specified the individual requirements of each qualification. The NZQA's website also provided the information needed to map the NQF structure that would be used to identify which subfield or domain that each unit standard achievement belongs. Using these data sources and the prototype report, the business rules that would facilitate the generation the reports required by the supervisor were collected. The process began by identifying the structural assertion type business rules associated with the NQF hierarchy. Object Role Modelling (ORM) constructs were used to articulate the structural properties of the framework and then these rules were presented to management and the supervisors for initial feedback. This provided an early opportunity to assess how meaningful these business rule expressions were to these domain experts before modelling other areas of the sub-system. Several examples of the structural assertion type business rules discovered are provided below.

- Field contains Subfield/ Subfield is within Field
- Subfield contains Domain/ Domain is within Subfield
- Domain contains Unit Standard/ Unit Standard is within Domain

Each business rule has been provided with a reverse reading (the phrase after the forward slash '/') and it can be seen that the approach allows the hierarchical structure of the framework to be articulated simply, without any reference to the eventual implementation technology. It should also be noted that no reference scheme (i.e. unique identifier) or constraints (i.e. optionality and cardinality) are indicated within

these verbalisations, although such details can be specified as each rule is entered into Visio Modeller. When presented with the above verbalisations, the domain experts indicated that they had no difficulty in comprehending the rules. They identified two major factors that contributed to their clarity:

1. A consensus had been reached on the meaning of each of the terms used.
2. The rules were articulated in simple English.

Based on these findings, the task of identifying and articulating the remaining business rules proceeded as planned. As this task progressed, familiar examples were chosen from the NZQA website that was used to populate the Fact Tables in Visio Modeller. These examples not only provided instances of each business rule that would aid domain experts in their validation during the action research phase of the project, but also defined the uniqueness constraints (i.e. cardinality) that apply to each rule. Using the terminology of the Business Rule Group (BRG), such constraints are referred to as a type of action assertion. Thus, care was taken to ensure that the examples selected were significant in terms of being able to accurately represent the action assertions that would place constraints on the structures defined by structural assertions. All of the business rules discovered within the PTE and from the NZQA website are documented in Appendix 5.

6.3.9 Participation of Domain Experts in the Definition of Business Rules

Due to the comprehensive nature of the information provided on the NZQA website, it was not considered necessary to involve domain experts during the collection of rules sourced from this site. As such, the domain experts' involvement, at least as far

as these rules were concerned, was limited to their validation during the action research component of this project. However, domain expert involvement in the collection of rules may be necessary in other situations where they represent the main or sole source of those business rules. In this case study, domain experts were involved in discovery the business rules in relation to student assessment grouping.

The inclusion of assessment groups in the requirements for the new system was based on the need to 'bundle' a student's achievements into more than one group. This would allow supervisors to match subsets of a student's total achievements against the requirements of qualifications, since unit standards achievements may be used to satisfy the requirements of more than one qualification. The feature also allowed the achievements of any number of students to be grouped together so that the performance of students associated with a particular supervisor could be monitored by management.

This feature of the new sub-system has no parallels in the existing system. It is also specific to the needs of the PTE and as such, required the participation of domain experts to capture the rules that would define the required structural properties and the constraints that would apply to those structures. The inclusion of this feature also provided an opportunity to use Visio Modeller with the domain experts to assess this tool's efficacy in defining business rules in-situ.

Defining business rules with a domain expert alongside an analyst using Visio Modeller proved to be an effective way of modelling the problem domain. Initially, there was a sense of some scepticism from the domain expert who believed their

contribution to such a "technical" exercise would be of limited value. However, as activity progressed, the domain expert was able to witness immediately how the information they provided resulted in the creation of business rules that, "actually make sense!" A further benefit of working with the domain expert in this way is that the analyst is able to receive immediate feedback from them as the questions that naturally arise from this activity become apparent.

6.4 THE FINDINGS OF THE ACTION RESEARCH

As a consequence of the desire to involve domain experts in the validation of the requirements of the new mentoring sub-system, action research was adopted as the method applied during this activity. This collaborative approach allowed the dual goals of action research to be met; it provided the basis for testing one of the assertions of this thesis, which is that the quality of system requirements may be improved by involving end-users in their validation and, in doing so, resulted in an application that more closely meets the needs of those end-users. However, to test this assertion, it was necessary to conduct a second case study that critically appraised the sub-system whose requirements were validated during this action research component. The findings of this second case study are discussed in section 6.5. The immediate focus of this discussion will now centre on the findings of the action research.

6.4.1 Using Examples for Validating Constraints

After defining the basic structural assertion type business rules, examples were provided to populate the Fact Table of each rule. These details are used by Visio Modeller to apply a corresponding uniqueness constraint (i.e. cardinality) on the role played by each object within a rule. Using the terminology of the Business Rules Group (BRG), populating Fact Tables with examples allows a category of Action Assertion known as Integrity Constraint to be defined[1]. Many of these examples were drawn from the NZQA website during the first case study (New Zealand Qualifications Authority, 2004).

The main thrust of the action research component was to involve end-users in the validation of these examples. However, as described in section 6.3.9, the requirements for the new system included the ability to define student assessment groups. Since this was a new feature, there were no examples that were already familiar to the domain expert. In section 6.3.4, a lack of such familiar examples was cited as a possible problem, especially if the active participation of domain experts in validation of business rules is sought. Thus, the first task in this action research component was to investigate this problem with domain experts so as to determine an approach for resolving such difficulties.

The domain experts who required the assessment grouping feature, worked in collaboration with the researcher and Visio Modeller. It was discovered during this activity that it would be very easy for the analyst to steer the domain expert into confirming what the analyst personally believed to be true. In an attempt to avoid this

[1] Refer to section 5.7.2 for a full description of Integrity Constraints

situation, it was found useful to provide domain experts with *sets* of examples. These example sets expressed all the possible uniqueness constraints that could be meaningfully applied to the basic structural details of assessment groups. Since all forms of constraints can be verbalised automatically by Visio, this approach allowed them to select the example set that generated a verbalisation that most accurately described how each structural assertion should be constrained. In most situations, allowing the domain expert to select the most appropriate example set simply confirmed the pre-conceptions of the researcher. That is, they chose the example set that the researcher would have selected had the domain expert not been involved. Occasionally, this did not occur and in most cases, these situations were the result of some confusion in the minds of the domain experts.

The confusion experienced by the domain experts with some example sets, appeared to derive from the reference scheme (i.e. unique identifier) used to identify instances of each object. When populating a Fact Table in Visio, values are provided that represent instances of the reference scheme associated with each object. Table 6.1 documents the verbalisations that initially caused some confusion for domain experts, as a result of using numerical values to populate the corresponding Fact Table.

Domain 1 is within Subfield 1
Domain 2 is within Subfield 1
Domain 3 is within Subfield 2

Table 6.1 – Verbalisations Resulting from a Numerical Reference Scheme

Although using numbers for a reference scheme is a reasonable approach, since such values are likely to be used to identify rows in the resulting database tables, they do

not assist the domain expert in their attempt to conceptualise constraints. One would expect this problem to be more apparent when modelling concepts that previously did not exist within the system, simply because they are unfamiliar to the domain expert. However, it could be a more generic problem in that domain experts would tend to mentally assign a name to instances of an object, rather than a number. For example, a supervisor would almost certainly recall the name of a student when they meet them for a mentoring session, rather than the student number that has been assigned to them. This is also most likely the case for many other types of objects. For example, the name of a qualification or programme offered by the PTE would be more meaningful to most of its employees than the reference number assigned to these concepts. However, one would expect that other concepts, such as a student's enrolment form, to be associated more with numbers than names. Nevertheless, in view of this finding, it was decided that Fact Tables, wherever possible and meaningful to do so, would be populated with the names associated with instances of an object rather than numeric reference schemes. As an example of this approach, Table 6.2 has been reproduced below in which instances of each object have been identified using appropriate textual descriptions, rather than numbers.

Domain 'generic computing' is within Subfield 'Computing'
Domain 'computer support' is within Subfield 'Computing'
Domain 'work and study skills' is within Subfield 'Core Generic'

Table 6.2 – Verbalisations Resulting from a Textual Reference Schemes

This approach appeared to aid the domain experts in their validation of the examples and resulted in fewer calls for clarification. One domain expert said, "It [the verbalised example] reads much better because I don't need to remember what the

numbers stand for". It could also be argued that adopting textual reference schemes where possible is a more conceptual approach to the specification of constraints. It is not necessary to define numerical reference schemes simply because the unique identifier in the resulting database table will use such a format. As long analysts are mindful in selecting textual reference schemes that also exhibit the desired characteristics in terms of the uniqueness constraints they express, the aims of both clarity and technical accuracy can be met.

Although examples within Fact Tables enable the analyst to define the cardinality of roles between objects, constraints such as optionality and other more complex categories of action assertions, required other techniques. The findings associated with the validation of these constraints are discussed in the next two sections.

6.4.2 Validating Optionality and Cardinality for Binary Predicates

The optionality of each role within binary predicate is defined in Visio Modeller by selecting the most appropriate constraint option from a list. The option selected is used to answer a constraint question involving each role. There are four options available; 'Zero or One', 'Zero or More', 'Exactly One' or 'One or More'. Consequently, both optionality and cardinality details may be specified by responding to these questions, as shown figure 6.1 In this example, the optionality details for the fact 'Domain is within / contains Subfield' have been specified by answering two constraint questions: 'Each Domain is within how many Subfield' and 'Each Subfield contains how many Domain'.

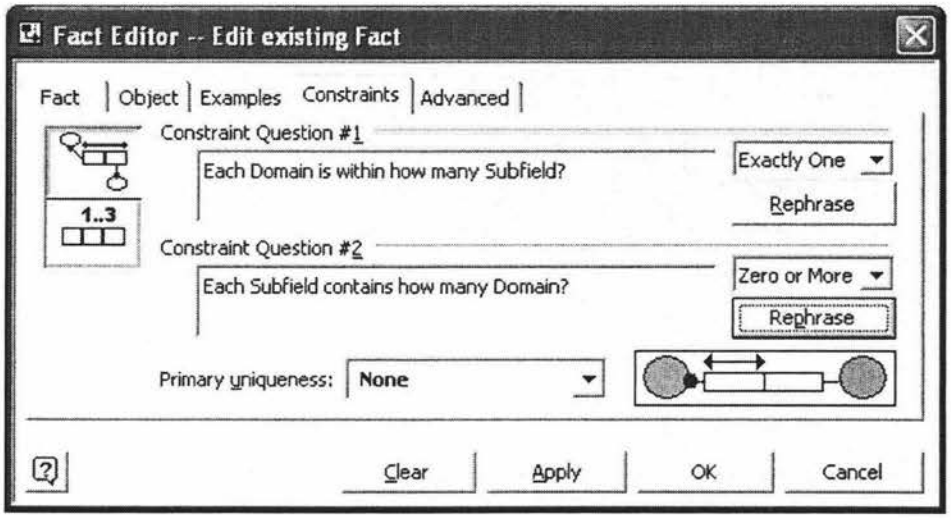


Figure 6.1 – Specifying Basic Constraints in Visio Modeller

Based on the selection made by the analyst, the constraint is verbalised automatically by the tool. Using the options selected in figure 6.1, the following verbalisations will be generated.

> 'Each Domain is within some Subfield'
> 'Each Domain is within at most one Subfield'

The first verbalisation specifies that every Domain must be associated with a Subfield and thus each Domain plays a mandatory role with Subfield. The second verbalisation asserts that every Domain is within only one Subfield, thus establishing the cardinality of the role. Since a Subfield can contain zero or more Domains, there is no uniqueness constraint or mandatory role constraint that requires verbalisation.

For business rules involving a binary predicate, most domain experts had little difficulty in understanding either the constraint questions or the verbalisations that resulted by answering these questions. However, it was occasionally recognised that some domain experts required further clarification in relation to the verbalised optionality constraint. Using the above example, 'Each domain is within some

Subfield', expresses the constraint that a Domain plays a mandatory role with Subfield. The word '*is within*' however, was perceived as a little 'weak' by several domain experts in conveying this constraint. Modifying the verbalisation as follows 'Each Domain *must be* within some Subfield' was perceived as a much more explicit and immediately obvious expression of such constraints.

6.4.3 Validating Constraints for Ternary Predicates

In the above example, the basic structural assertion involved a binary predicate. However, several business rules involving a ternary predicate were also discovered within the Private Training Enterprise (PTE). Establishing the constraints that apply to these objects requires a somewhat different technique, as illustrated in the following example.

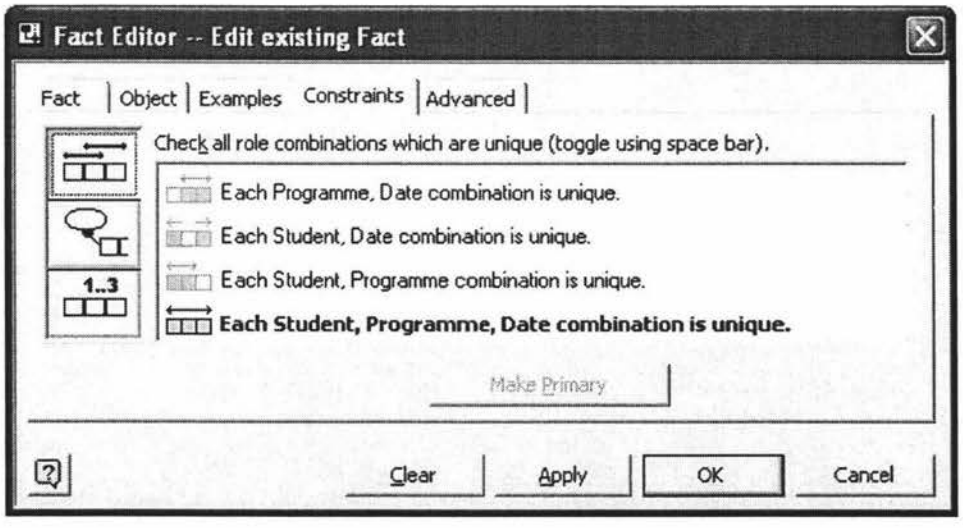


Figure 6.2 – Defining Cardinality within Ternary Predicates

In figure 6.2, the options in terms the uniqueness constraints (i.e. cardinality) that may be applied to the basic business rule 'Student withdraws from Programme on Date', are

shown. The analyst selects the most appropriate unique role combination(s) from this list and the resulting verbalisation is displayed automatically. Using the example above, the role combination 'Student', 'Programme' and 'Date' has been identified as unique and would result in the following verbalisation.

'It is possible that more than one Student withdraws from more than one Programme on more than one Date'.

Verbalisations such as the one above, did present several domain experts with problems if an attempt was made to use them in isolation. However, by augmenting such verbalisations with examples that conform to the uniqueness constraint(s), the meaning of the constraint(s) appeared to be much more transparent to these people. The examples used to result in the above verbalisation are as follows[2].

1. Student 'Julie Edmonds' withdraws from Programme 'Youth Training' on Date 27/03/2003.
2. Student 'Julie Edmonds' withdraws from Programme 'Youth Training' on Date 15/06/2004.
3. Student 'Julie Edmonds' withdraws from Programme 'Training Opportunities' on Date 15/06/2004.
4. Student 'Phillip Steeton' withdraws from Programme 'Youth Training' on Date 15/06/2004.

Even so, it was found that the significance of each example in its relation the resulting verbalised constraint had to be explained to most domain experts. However, if analysts take the time to perform this task, domain experts were able to provide valuable assistance in the validation of constraints of this type.

[2] The student names used in this and all other verbalisations are fictitious.

Specifying mandatory role constraints (i.e. optionality) for ternary predicates involves a similar procedure to that described above, in that analysts select the most appropriate constraint from a list of options. In the example below, the constraint option that every instance of 'Elective' must participate in the fact 'Elective has Minimum Credit Value at Level', has been selected.

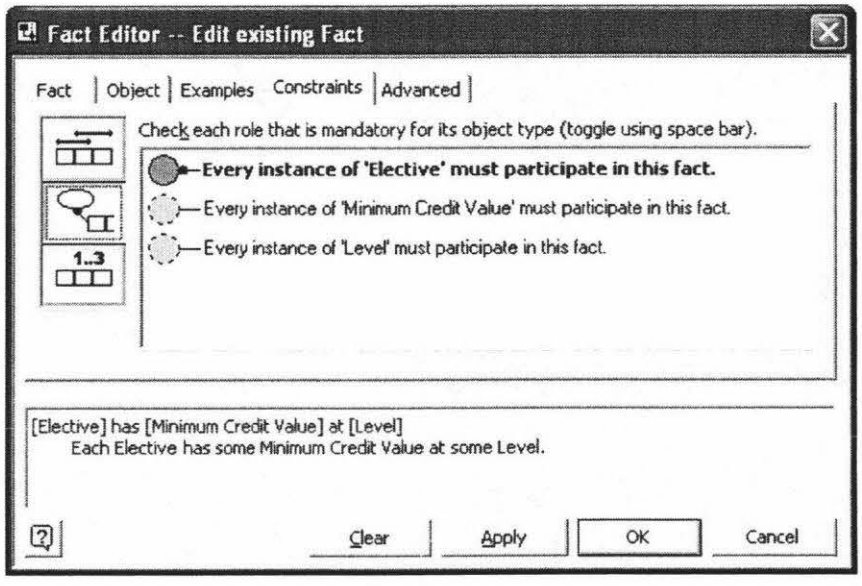


Figure 6.3 – Defining Optionality in Ternary Predicates

Selecting this option results in the following verbalisation:

'Each Elective has some Minimum Credit Value at some Level.'

Again, it was found that domain experts preferred a more explicit declaration of such constraints, as documented in section 6.4.2. Applying this approach in the above example, the verbalisation is modified to the following:

'Each Elective *must have* some Minimum Credit Value at some Level.'

6.4.4 Validating Complex Constraints

In this section, the experiences of the researcher and domain experts in the validation of more complex constraints involving set comparisons, disjunctive mandatory roles and textual rules are discussed.

In the example below, a Subset constraint (indicated as by the ⊆ symbol) has been defined that asserts that the set of Students withdrawing from a Programme is a subset of Students enrolling on that Programme. This a form of set comparison constraint that falls into the Integrity Constraint class of Action Assertion type business rules, as described in section 5.7.7 of Chapter 5.

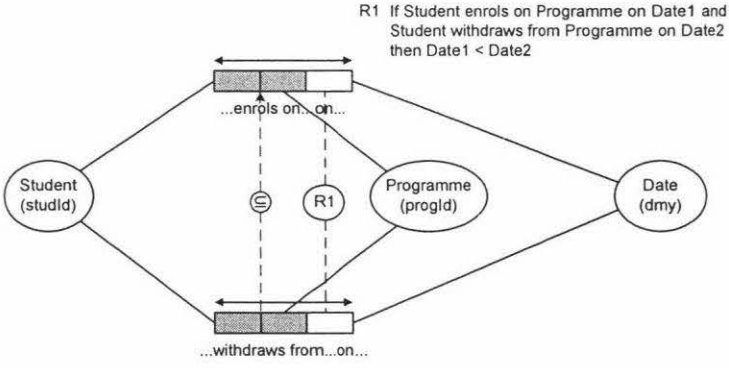


Figure 6.4 – Validating Subset Constraints

Using Visio Modeller, the analyst simply identifies the role boxes of each predicate involved in the constraint (indicated by shading) and selects the appropriate set comparison. Based on the model in figure 6.4, the following verbalisation will be automatically generated.

'If Student s withdraws from Programme p on some Date then

Student s enrols on Programme p on some Date.'

In section 5.7.8, it was suggested that the use of algebraic symbols within verbalisations could be confusing to some domain experts. However, by simply replacing these symbols with values taken from the Fact Tables, the readability of the resulting verbalisation maybe improved as illustrated below.

'If Student 'Julie Edmonds' withdraws from Programme 'Youth Training' on some Date then Student 'Julie Edmonds' enrols on Programme 'Youth Training' on some Date.'

Domain experts reported that they found such verbalisations to be a 'very formal' way of saying something quite simple. When shown the above verbalisation, one domain expert suggested that the phrase, 'A student can only withdraw from a programme if they've previously enrolled on it', was a more natural way of conveying the same information. This observed formality, is of course a consequence of the mathematical framework that underpins ORM constructs (Halpin, 2001) and the restrictive grammar that is used to express those constructs. Despite this observation, the domain expert obviously understood the verbalisation and had inadvertently identified an additional constraint.

This constraint exists on the values that may populate the 'Date' object, indicated in figure 6.4 as the circled 'R1' symbol. Using ORM terminology, the constraint 'R1' is a form of textual rule, which are described in section 5.7.17. The conditional nature of this particular textual rule identifies it as a Condition Constraint class of Action Assertion type business rule. No form of textual rule can be implemented automatically in the current version of Visio Modeller, but analysts are able to document such rules as a textual annotation using ORM's restrictive grammar. The annotation associated with the textual rule 'R1' in figure 6.4, is as follows.

'If Student enrols on Programme on Date1 and Student withdraws from Programme on Date2 then Date1 < Date2.'

When domain experts were presented with the above verbalisation, most required only a brief explanation to understand the assertion and its associated condition. However, they did not appreciate why this rule could not be combined with the equality constraint, described earlier in this section. In this sense, the transparency of the validation process was obscured somewhat by domain expert's lack of knowledge of ORM constructs. Despite this difficulty, they were prepared to accept that 'this is the way such details are specified', even if they generally felt it was 'not the most natural way' to convey such information.

The final category of complex constraint validated by domain experts relates to a disjunctive mandatory role (inclusive-or), described in section 5.7.10. This constraint was applied to the requirements of Qualification electives and is graphically depicted in figure 6.5. A disjunctive mandatory role asserts that instances of an object *must* participate in at least one of the roles between which the constraint is applied and, as such, is an Integrity Constraint class of business rule.

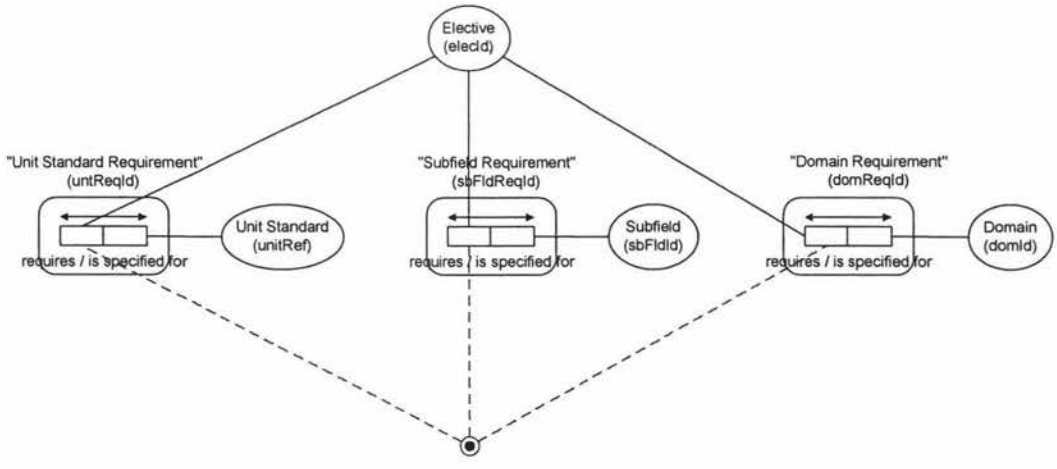


Figure 6.5 – Validating Disjunctive Mandatory Roles

In the above example, Visio Modeller verbalised the constraint as follows.

'Each Elective requires some Domain or requires some Subfield or requires some Unit Standard.'

As in logic, the implied assumption is that 'or' is always interpreted in the inclusive sense. Thus, in the above verbalisation an Elective could require a Domain, Subfield, Unit Standard or possibly all three. However, without explicitly stating this assumption, several domain experts interpreted the use of the 'or' in the exclusive sense, when presented with the above verbalisation. This problem was easily overcome by modifying the verbalisation as follows.

'Each Elective requires some Domain or requires some Subfield or requires some Unit Standard or requires some combination of Domain, Subfield or Unit Standard.'

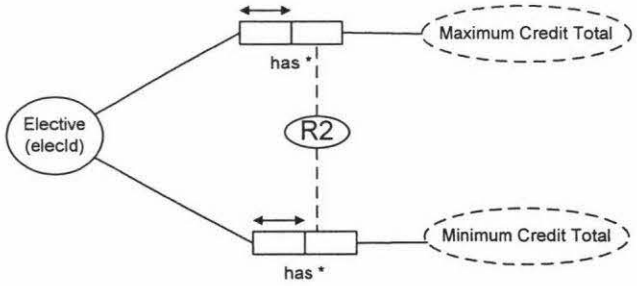
Although these difficulties were easily overcome, it should be appreciated that even slightly modified verbalisations cannot be used to generate logical designs since the mechanism used in this translation requires unadulterated FORML statements. This problem was also easily circumvented by simply copying the verbalisation into a word processor and modifying it accordingly before presenting it to the domain experts. If a consensus on the validity of the modified verbalisation was reached with domain experts, the original verbalisation within Visio Modeller was left unchanged. Similarly, if the modified verbalisation proved to be inappropriate in accurately capturing a business rule, then Visio Modeller was used to generate an alternative verbalisation based on the feedback from domain experts. This process of generating verbalisations, modifying them for clarity within a word processor and validating them with domain experts, continued until the appropriate verbalisation was

identified. This process ensured that the unmodified version of the verbalisation was always available in Visio Modeller to generate the logical schema while still accommodating the need for clarity with domain experts.

6.4.5 Validating Derivation Rules

As described in section 5.8, derivation rules are applied to an ORM schema when the value of object instances may be calculated from other values. In such situations, it is normal practice to have these values calculated automatically by the system, thus reducing the possibility of human error.

In the example below, the value of both Maximum Credit Total and Minimum Credit Total can be derived by determining the sum of the values associated with other objects. Maximum Credit Total is the maximum total credits across all levels allowed from a particular Qualification's Elective. Minimum Credit Total is the minimum total credits across all levels allowed from a particular Qualification's Elective. Since the minimum and maximum number of required credits at each level is recorded by the system, the minimum and maximum credit totals across all levels may be calculated by simple arithmetic.



* Minimum Credit Total = Sum of each Minimum Credit Value at Level
* Maximum Credit Total = Sum of each Maximum Credit Value at Level

R2 If Elective has Minimum Credit Total and Elective has Maximum Credit Total then Maximum Credit Total > Minimum Credit Total

Figure 6.6 – Validating Derivations

When instances of an object are to be derived, an asterisk is placed on the model adjacent the role box of the derived fact. Using ORM's the restricted grammar FORML (Formal Object Role Modelling Language) the derivation rule is expressed as a textual annotation on the model. In the above example, these are as follows.

'Minimum Credit Total = Sum of each Minimum Credit Value at Level'
'Maximum Credit Total = Sum of each Maximum Credit Value at Level'

Although domain experts had no difficulty in validating these simple derivation rule details, no attempt was made to validate other, more complex rules. In these situations, ORM allows analysts to express rules both formally and informally to facilitate their validation by domain experts. However, this was not necessary within the domain examined by this research. As such, no conclusions can be drawn as to the efficacy of employing ORM to express and validate more complex derivations.

The current version of Visio Modeller cannot automatically implement derivations from the conceptual model into the target DBMS. However, Visio Modeller does allow analysts to document derivations as comments that could be used to annotate the conceptual model. These comments can then be used as the basis for manual implementation of derivations within the logical model. Since derivations are modelled as unrestricted comments, analysts are free to employ any formal or informal language to express such rules, including Structured English. This would allow programming constructs such as repetition and selection to be incorporated into the definition of a complex rule. However, just how appropriate such constructs are when domain experts are involved in their validation is debatable. Ideally, even complex rules should be expressed at the conceptual level using constructs familiar to

typical domain experts. This suggests that other conceptual techniques may need to be developed to express complex algorithms to domain experts. Finally, although examples are usually restricted to populating Fact Tables, they could also be employed to demonstrate the application of derivation rules to domain experts.
The textual rule, indicated graphically as the circled "R2" symbol in figure 6.6 and verbalised below, is associated with the derivation rule previously described. It simply asserts that the Maximum Credit Total for an Elective must exceed the Minimum Credit Total for that Elective. Examples could have been provided to demonstrate the validity of this assertion, but domain experts did not feel this was necessary for this particular rule.

R2 If Elective has Minimum Credit Total and Elective has Maximum Credit Total then Maximum Credit Total > Minimum Credit Total

6.4.6 Action Research Conclusions

The purpose of the action research component was to validate the PTE's business rules in a collaborative effort with domain experts. These business rules were used to define the requirements of the organisation's student mentoring sub-system. Thus, this research provided the basis upon which to evaluate the efficacy of employing ORM constructs, to express rules in form that allows domain experts to challenge the researcher's perceptions concerning this new sub-system. This evaluation is conducted in section 6.5.2.

Action research proved to be an effective tool with which to define the requirements of the PTE's new sub-system, since the collaborative nature of the approach fostered

an atmosphere in which domain experts played a central role. Synthesizing Business Rules Modelling with ORM concepts to express requirements helped to further emphasize the importance of the role they played, by providing a conceptual framework to discuss and challenge the perceptions of the researcher. The findings of the action research component of this thesis indicate that ORM's ability to verbalise all categories of business rules does indeed improve the transparency of analysis activities for domain experts. By improving the transparency with which data requirements are gathered, it is suggested that domain experts are more able to be actively involved in the validation of these requirements. Based on these findings, the case for defining a system's data requirements in the form of business rules expressed using ORM constructs are summarised below.

1. Business rules originate from domain experts and so are meaningful to them.
2. Business rule concepts are also familiar to system analysts, thus providing a common framework for discussing data requirements.
3. ORM constructs provide analysts with a powerful tool for expressing all categories of business rules, as defined by the Business Rules Group.
4. All ORM constructs can be verbalised in the restrictive grammar of FORML that generally allows domain experts to become actively in the validation of data requirements.
5. Examples in the form of Fact Tables further promote the transparency of business rule expressions to domain experts, which should lead to greater accuracy in the definition of data requirements.

Despite these encouraging findings, the action research component did identify areas where ORM constructs caused some confusion in the minds of domain experts. These areas are summarised overleaf, together with the approaches taken to overcome these difficulties.

Problem Area	Problem Solution
Numeric reference schemes within Fact Tables often do not assist domain experts to conceptualise constraints.	Adopt textual reference schemes wherever possible and meaningful to do so.
The verbalisations expressing the cardinality of ternary predicates were often difficult for domain experts to fully appreciate.	Illustrate these constraints with examples from the Fact Tables taking care to emphasize the significance of each example on the resulting verbalisation.
Complex constraints such as set comparisons are verbalised using algebraic symbols that confused some domain experts.	Modify the verbalisation by replacing algebraic symbols with actual values from the Fact Tables.
The verbalisations expressing the optionality of ternary predicates were not considered to be sufficiently explicit by several domain experts.	Modify the verbalisation to include '*must have* some' to express a mandatory role constraint.

Table 6.3 – Identified Problems and Solutions when Applying ORM

What was particularly satisfying for the researcher was that despite the difficulties encountered when applying ORM constructs to express business rules, each problem had a satisfactory solution. Often, this solution required nothing more than to make greater use of the values populating the Fact Tables. In other situations, only slight modifications to the verbalisations automatically generated by Visio Modeller were required in order to improve their clarity.

6.5 THE FINDINGS OF THE SECOND CASE STUDY

The aim of the second case study was to determine the efficacy of expressing business in a form that facilitates their validation by domain experts. From a technical perspective, Chapter 5 demonstrated that ORM constructs are able to express all categories of these rules. The findings of the action research suggest that business

rules are meaningful to domain experts and that ORM constructs provide an effective language to express these rules. A question remains however, as to whether the synthesis of these two modelling approaches can improve the quality of requirements.

It has been argued[3] that improving the transparency in which requirements are specified, fosters a more collaborative effort between the analyst and the domain expert. If, as a consequence, domain experts are more able to challenge the analyst's perception of their system, there should be a greater likelihood that the resulting system will meet the needs of the end-user community.

To test this assertion the second case study was conducted within the target organisation, soon after the implementation of the student mentoring sub-system. As described in section 6.3 and 6.4, the requirements for this sub-system were specified using ORM's Formal Object Role Modelling Language (FORML) and validated by domain experts. In the next section, how these requirements were applied in the implementation of this sub-system is briefly described and will be followed by a discussion as to how the resulting application was evaluated.

6.5.1 Conceptual Design to System Implementation

Having defined the requirements for the new mentoring sub-system and validated these details with domain experts, a logical model (see Appendix 7) was then generated automatically from within Visio Modeller. The tool is able to generate a wide range of relational schemas including MS SQL Server, the target DBMS of the Private Training Enterprise (PTE). Although, as yet, no provision has been made for

[3] Refer to section 4.2.2

mapping ORM schemas into Object Orientated environments, such support is expected in future releases of the tool. Visio Modeller transforms ORM conceptual schemas using an algorithm that generates table definitions in Optimal Normal Form (ONF), which is at least equivalent for 5NF (Halpin, 2001). Database designers are then able modify the results of this mapping algorithm before generating the database from that design.

Using the logical model generated from the conceptual schema, the PTE's application developer modified the default design for the student mentoring system so that it conformed to developer's personal preference of a simple key for each table. The developer has a strong dislike for composite primary keys, particularly if there is even a remote chance that the value of one or more of those keys could change over the lifetime of a record. Other changes included the abbreviation of some of the table names, automatically generated by the tool. Once these modifications were complete, the logical design was used by Visio Modeller to generate a script file (see Appendix 8). The file contains all the code necessary to build the database tables, indexes and constraints, as defined in the logical design. The file was run in a test environment that contained copies of all tables from the existing system. These tables were consolidated with the newly generated tables that were then populated with test data. The final step was to construct the reporting mechanisms that would extract the data from the tables and present these details in the format specified by the domain experts (shown in Appendix 3). As described in section 6.3.5, the organisation's supervisors believed that a reporting feature that documented the requirements of a qualification and the student's achievements and options towards its completion, would be an effective way of conveying the information they needed to guide their students. MS

Crystal Reports was chosen as the environment in which these reports were created by the PTE's application developer, since this tool has been used to develop reports for all of the organisations sub-systems.

6.5.2 Evaluating the Quality of System Requirements

The first case study revealed that students who enrol on programmes at the PTE, work towards the completion of either the National Certificate in Computing (Level 2 or 3) or the National Certificate in Business Administration (Level 2 or 3). In view of this finding, the reporting mechanisms constructed for these qualifications were used as the basis for evaluating the quality of the system requirements that lead to their creation.

It is recognised that by focusing purely on the output generated by the system, the evaluation ignored issues relating to the user-interface that are generally considered crucial to the success of systems. However, since the dominant themes of this thesis concerns issues relating to the specification and validation of data requirements[4], the quality of the user-interface was considered to be beyond the scope of this evaluation.

The criteria used to evaluate the quality of the data requirements are documented below.

1. The specified content of the reports can be extracted or derived from the database tables.
2. The data integrity constraints, captured during the action research, are enforced during data entry.
3. The structure of the database allows the requirements of the qualifications currently offered by the PTE to be recorded.

[4] Refer to section 4.2.2

4. Student achievements can be matched against a qualification's requirements to determine whether a student has met the requirements for that qualification.
5. The options available to a student, in terms of unit standard achievements that may be credited towards a qualification, can be determined from data held in the database.
6. The structure of the database allows the requirements of other qualifications, not currently offered by the PTE, to be recorded.

Obviously, only a direct comparison involving the creation of the same system using the ORM/business rules approach in one instance and a traditional approach in another, can ultimately determine if improvements in quality are a realisable goal. Unfortunately, pragmatism and time constraints precluded this ideal. However, it is believed that if the mentoring sub-system adequately addresses each of the above quality criteria without the need for significant modification to the database design, the approach advocated within this thesis is at least worthy of further research.

All criteria except item 2 in the above list were arrived at after discussions with a small group consisting of both management and domain experts. Of high importance to this group was that the reporting mechanisms, and the database used by these reports, were capable of generating the details specified in the prototype report design (see Appendix 3). This high level requirement was included in the list of criteria as item 1, but to achieve this goal, the lower level requirements identified as items 3,4 and 5 must also be met. Item 6 in the above list identifies a need for flexibility in the design of the database, so that it can meet the future needs of the organisation in its delivery of qualifications. Finally, item 2 was a criteria added by the researcher to ensure issues relating to data integrity were also addressed.

The first evaluation criteria that was applied against the database design was item 2, since it made sense to assess data integrity as the newly created database tables were

populated with data. In addition to primary and foreign key constraints, the database design also included uniqueness and value constraints on a number of table columns and functions that enforced equality constraints. Primary, foreign key and mandatory role constraints were tested simply by populating the database tables with the test data that were used to construct the reports. However, the test data also included values that were chosen to deliberately fail the integrity tests associated with each primary and foreign key constraint. Attempts were also made to record null values in those columns associated with a mandatory role constraint. The DDL statements generated automatically by Visio Modeller were found to enforce these constraints correctly in all cases.

A similar procedure was used to ensure that the column uniqueness, value and equality constraints that were applied to some tables were also enforced. Using a combination of valid and invalid test data, these constraints were tested to ensure that they enforced data integrity. Value constraints were tested by providing data values within and outside of the specified number range or set of defined valid values. Uniqueness constraints, which were often applied to a number of columns within the same table, were tested to ensure that uniqueness was enforced not only for single columns, but also across combinations of columns, where appropriate. Finally, set comparison constraints were tested to ensure that they were applied accurately. As a consequence of conducting these tests, the DDL statements generated by Visio Modeller proved to be sufficient in ensuring the integrity of the data.

By addressing item 2 from the list of evaluation criteria identified earlier in this section, the database was populated with the data necessary to address the remaining

items with the exception of item 6. Reports that documented the achievements of a number of fictitious students and their progression towards the completion of the qualifications currently offered by the PTE, were generated and presented to the evaluation group (see Appendix 6). The group was invited to compare these reports against the original prototype (Appendix 3) created in MS Word and to make comments. As can be seen by comparing Appendix 3 with Appendix 6, a high degree of conformity in terms of the content of the reports with the original specification has been achieved. This was reflected in the comments made by the evaluation group who were generally pleased with the generated reports. In particular, they were impressed by the documentation of the options available to students in terms of the unit standards that may be credited towards a qualification. Although this was stipulated as one of the original requirements of the mentoring sub-system, supervisors were very pleased to observe that it was a requirement that had been met in the eventual sub-system. One supervisor commented that 'listing the [unit standard] options will save us a lot of time'. Supervisors previously had to browse through the NZQA web-site to determine which subfield and/or domain a unit standard belonged to together with its level and credit value in order to determine its applicability in meeting a qualification requirement.

The evaluation group identified one problem area within the generated reports. They believed that the layout of the reports associated with the National Certificate in Computing (NCC) should be modified so that the requirements of each elective and a student's progression towards meeting them are documented in the summary section, at the head of each of these reports. Originally, the requirements were summarised at the level of the qualification as a whole in terms of compulsory, subfield and domain

requirements. However, for NCC qualifications, students must satisfy the requirements of each elective to satisfy the requirements for the qualification as a whole. Consequently, supervisors believed that student progression details at the level of the elective were needed to adequately mentor students. Fortunately, this additional requirement did not result in any modifications to the database itself since these details were already being used by the reporting programs, but simply not displayed. As a consequence, addressing this issue only required a change to the format of the reports associated with the NCC qualification and these changes are reflected in reports provided in Appendix 6.

Having addressed items 1 to 5 in the list of evaluation criteria with the end-user group, the final criteria, item 6, became the focus of the group's attention. This item assessed the ability of the data model to support the recording of new qualification requirements, which would be needed if the organisation expands its current provision of programmes. Management also felt that there would be an increasing need to be able to map Achievement Standards, in addition to Unit Standards, as more school leavers enter its programmes with credits from this national standard. Achievement Standards are studied by students still within the compulsory education system and are usually credited towards the National Certificate in Educational Achievement (NCEA). However, Achievement Standards can also be credited towards the completion of qualifications on the NQF in the same way as Unit Standards. Since Achievement Standards are simply a subset of the 18,000 Unit Standards currently registered on the NQF (New Zealand Qualifications Authority, 2004), mapping these standards into the table structure of the new sub-system requires no changes to that structure. Achievement Standards are also designated with the same reference number

as their Unit Standard equivalent and can therefore, be treated exactly as though they are those Unit Standards. This was explained to management who were satisfied that the new sub-system addressed this requirement.

The main focus of item 6 on the list of evaluation criteria was to determine the flexibility of the new sub-system's schema to support the provision of other qualifications, not currently offered by the organisation. Since the NQF contains over 800 registered qualifications (New Zealand Qualifications Authority, 2004), it was obviously impractical to determine whether the requirements of each of these can be mapped into the new table structure. Indeed, the requirements of qualifications associated with one Field in the NQF are often significantly different in terms of their structural properties compared to qualification requirements in other fields. This is also often the case for qualifications within the same Subfield or Domain that are offered at different levels. For example, the National Diploma in Computing (level 5 to 7) has optional specialist strands with specific requirements. However, the National Certificate in Computing (levels 2 to 4) has no optional strands. This difference in structure was dealt with by simply treating each strand as an optional elective. This was possible since strands have requirements whose structural properties are same as they are for electives. Unfortunately, structural differences noted in many other qualifications during the case study could not be dealt with as easily, since they conflicted with the structural properties of the requirements of the qualifications that the organisation currently offers. These differences have arisen because qualification standard setters are comprised of a diverse range of expert groups, who have established requirements for specific qualifications independently of other groups (New Zealand Qualifications Authority, 2004). Thus, although many common

structural patterns exist across all qualifications on the NQF, significant and often conflicting patterns have also emerged over time. Consequently, after consultation with management, the data model for the new sub-system was restricted to address the data requirements of qualifications currently offered and those that management anticipated that they would offer within the next 2 years.

The original intention was that the organisation's system developer would construct similar reports to those created to address evaluation items 2 to 5. However, the developer considered this unnecessary since the ability to simply record the requirements of anticipated qualification offerings within the new schema, would effectively determine whether the schema addressed evaluation item 6. The reports simply present the extracted data within a predefined format. Management agreed with this rationale and so testing was restricted to the populating of the database tables with each anticipated qualification offering. These included the National Certificate in Computing (level 4), the National Diploma in Computing (level 5), the National Certificate in Business Administration (level 4) and the National Certificate in Public Sector Services (level 4).

After successfully populating the database tables with the requirements of these anticipated qualification offerings, the system developer concluded that the new schema addresses evaluation item 6 in a restricted sense. The developer reported this observation to the end-user group who were satisfied with this assessment, thus completing the evaluation of the new sub-system.

6.6 CONCLUSIONS

This chapter has presented the findings of both case studies and the action research component of this thesis. The initial case study aimed to determine the basic business rules (i.e. the structural assertions) of the target organisation. These business rules were then validated during the action research component in a collaborative effort with end-users, using examples to populate the Fact Tables associated with each rule. Providing these examples not only confirmed the structural assertions defined during the initial case study, but also allowed the capture of action assertions (constraints) that were applied to these structural details. Finally, a second case study was conducted in order to determine the efficacy of synthesizing ORM constructs with business rules concepts in an attempt to improve the quality of data-orientated system requirements.

When presented with the business rules captured during the initial case study, domain experts reported that they found them easy to comprehend and challenge. They attributed this quality to the fact that a comprehensive glossary of business terminology had been previously compiled, after a consensus had been reached as to the meaning of each of these terms. A further contributory factor was that these terms were associated to one another using simple English sentences to express structural assertions. The researcher also noted that domain experts were able to provide valuable and immediate feedback when they were directly involved in the articulation of business rules, as these were entered into Visio Modeller.

During the action research component of this thesis, example sets were used to populate the Fact Tables associated with each business rule. These examples not only confirmed the basic structural assertions captured during the initial case study, but also defined the action assertions (constraints) that applied to these rules. The researcher determined that the clarity of these action assertions could be improved by adopting textual reference schemes to identify instances of each object, wherever possible and meaningful to do so. Although textual reference schemes are unlikely to be applied in the final database design, within the confines of the conceptual model, both the aims of clarity and technical accuracy (in terms of defining cardinality and optionality details) can be met by adopting this approach.

The research also found that the clarity of action assertion type business rules can be further improved by making small modifications to the verbalisations automatically generated by Visio Modeller. These modifications included:

- Adding the phrase '*must be*' to the verbalisations that express mandatory roles (optionality) so that they more explicitly express such constraints.
- Replacing algebraic symbols used to express set comparison constraints with actual instances from the associated Fact Table.
- Explicitly linking examples drawn from the Fact Tables to the verbalisations associated with ternary predicates.

Textual rules are an ORM construct that may be used to express derivations, the final category of business rules as defined by the Business Rules Group (GUIDE Business Rules Project, 1997). Although domain experts found no real difficulty in validating the simple derivations that were defined for the mentoring sub-system, it is recognised that more complex derivations requiring logical constructs (i.e. repetition and

selection) could be much more challenging for domain experts to validate. Despite this potential problem, the findings of the action research component strongly suggest that ORM's ability to verbalise all categories of business rules, does promote a collaborative and more transparent approach to the definition of data requirements.

In order to assess whether the synthesis of ORM and business rule concepts has the potential to improve the quality of system requirements, a second case study was conducted. In this case study, a list of evaluation criteria was compiled with the aid of domain experts to appraise the mentoring sub-system. The reports generated from data extracted from the new database were compared against these criteria by management and end-users. The conclusion reached by domain experts and the researcher was that the mentoring sub-system satisfactorily met all of these criteria without the need for any significant changes to the database design. This conclusion suggests that the approach advocated within this thesis is capable of producing good quality database designs for reasonably complex systems.

Of course, a similar satisfactory outcome may have been possible using a more traditional approach to the specification of data requirements. However, the ability to express requirements in a form that may be challenged by domain experts allowed numerous misconceptions to be identified almost immediately, before the logical schema was generated. ORM constructs also allowed the constraints and derivations that applied to the data structures to be captured during analysis, thus adding to the completeness of the model produced. This capability is not shared by almost all data modelling techniques currently used in industry (Halpin, 2002), which suggests that ORM improved the quality of the data requirements of the mentoring sub-system.

CHAPTER 7

CONCLUSIONS

7.1 INTRODUCTION

In this final chapter, the conclusions drawn from the research undertaken are presented and the ramifications these have in relation to the hypothesis explored are examined. The limitations of the research and applicability of the findings within the IT industry as a whole are discussed and areas of future research that are suggested as a consequence of conducting this study are also considered.

7.2 CONCLUSIONS

The aim of this thesis was to suggest an alternative approach to the specification of data requirements in a form that facilitates their validation by domain experts. The approach involves a synthesis of two conceptual modelling techniques; Business Rules Modelling (as defined by the Business Rules Group) and Object Role Modelling (ORM). This synthesis allows business rules to be expressed using Formal Object Role Modelling Language (FORML) constructs; the restrictive natural language associated with ORM. Despite the mathematical framework that underpins this language, it was suggested that business rules expressed in FORML are significantly more comprehensible to domain experts than data requirements expressed using more traditional techniques. Consequently, there should be a greater

likelihood that those systems developed using the approach suggested in this thesis will conform to the needs of the end-user community.

In order to test this hypothesis, two case studies combined with action research were conducted within a small Private Training Enterprise (PTE). The findings of this research were presented in Chapter 6. The first case study aimed to capture the business rules that defined the structural details associated with a new student mentoring sub-system that was developed by the researcher for the PTE. Since these business rules related the terms employed within the organisation to express these structural details, it was suggested that the shared meaning of these terms should be determined and then documented within a glossary. It was found that producing a glossary of business terminology improved the transparency of the structural assertions that related these terms when they were presented to domain experts.

Visio Modeller, an automated tool, was used to construct ORM models that are able to graphically represent all categories business rules, as demonstrated in Chapter 5. This tool also allows assertions concerning the problem domain to be verbalised automatically within FORML statements, devoid of any reference to the eventual implementation technology. It was found that this facility enabled domain experts to comprehend and challenge the business rules captured by the researcher. Additional business rules were also sought from the New Zealand Qualification Authority's (NZQA) website. These details were required to facilitate the mapping of the National Qualification Framework (NQF) into the eventual database design of the new student mentoring sub-system.

The action research component aimed to validate the structural assertions captured during the first case study and to define the action assertions (constraints) and derivations that apply to these structural details. The process of validation involved a collaborative effort between the domain experts within the PTE and the researcher, using appropriate examples to populate Fact Tables within Visio Modeller. Examples not only aided the validation of the structural assertions but also allowed uniqueness constraints (optionality and cardinality) to be established. It was found that examples were essential when validating constraints that apply to structural assertions involving ternary predicates. These examples augmented the FORML statements automatically generated by Visio Modeller that domain experts found complex when used in isolation. Even so, it was found that the significance of each example in relation to the verbalised constraints on ternary predicates had to be carefully explained in order for domain experts to actively participate in their validation.

In most cases, verbalisations augmented with examples were sufficient to effectively engage domain experts in a collaborative approach for the validation of business rules. However, the verbalisations associated with more complex set comparison and mandatory role constraints (optionality) on ternary predicates did require minor modifications in order to improve their clarity. For example, the automatic verbalisation of set comparison constraints involved the use of algebraic symbols, which although simple, added unnecessary complexity. The clarity of such verbalisations was improved simply by replacing these algebraic symbols with an appropriate instance from the associated Fact Table. It was also suggested that textual, rather than numeric reference schemes (unique identifiers) be employed whenever possible and meaningful to do so. By adopting this approach, domain experts were not

required to mentally translate a code or number into the real-world object that such reference schemes are intended to identify.

The second case study aimed to test the assertion that by improving the transparency in which data requirements are specified, domain experts are more able to challenge the perceptions of the analysts who attempt to define those requirements. If this assertion is true, it suggests that systems developed using the approach advocated in this thesis should be more likely to conform to the needs of end-users. Conducting the first case study and more especially the action research component, this hypothesis seems to have merit. Providing domain experts with verbalisations and examples that describe and illustrate the business rules that exist in the PTE, does appear to foster a more collaborative approach based on a common understanding of the problem domain. However, in order to evaluate the quality of the requirements captured during the first case study and action research component, a second case study was conducted. This case study focused on establishing quality criteria against which the reporting mechanisms of the mentoring sub-system were then evaluated. The reports extracted data from the database whose requirements were defined during the first case study and action research component. These reports enabled student supervisors in the PTE to provide students with guidance as to the pathways available for the completion of qualifications on the National Qualification Framework (NQF). The conclusion reached by both domain experts and the researcher was that the mentoring sub-system satisfactorily met all of the established quality criteria. This suggests that the approach advocated within this thesis is capable of producing good quality database designs. Based on the findings of this thesis, the case for defining data

requirements in the form of business rules expressed using ORM constructs are summarised below.

- Business rules are meaningful to both domain experts and analysts and thus provide common framework with which to discuss problem domains and their data requirements.
- FORML statements can express all categories of business rules, as defined by the Business Rules Group, in a form that in general allows domain experts to challenge the perceptions of analysts. Using examples that define and illustrate business rules further promotes the transparency in which data requirements are expressed.
- Modelling the domain as a set of business rules expressed using ORM constructs seems to improve the completeness and quality of data requirements.

7.3 LIMITATIONS AND RELEVANCE OF THE RESEARCH

In this section, the limitations of the conducted research and the applicability of its findings for the IT industry as a whole are discussed.

7.3.1 Reliance on a Single Case

Although this research is not presented with a view of drawing generalisations that may be applied universally, its reliance on a single case could be criticised for limiting the relevance of the conclusions that have been presented. Ideally, a series of cases from a diverse range of domains would be required in order to establish a greater degree of certainty in the findings of this research. Quantitative analysis techniques may then have been applied to further the degree of confidence in the conclusions that

have been drawn. Unfortunately, the time required for a researcher working in isolation to investigate a series of cases precluded this option from being viable.

7.3.2 Limitations of the Suggested Approach

Using examples proved to be an effective approach for facilitating the active involvement of domain experts during the specification of the requirements for the new mentoring sub-system. However, difficulties could arise where examples familiar to the domain expert do not exist. This could occur in instances where an entirely new automated system is required, which is not based on any pre-existing manual system. Such situations could arise when an organisation enters a new area of business activity. One strategy that may be useful in such circumstances would be to investigate other similar systems recognised as incorporating best practice. Prototyping may also offer some insights as to the outputs required from a new system and thus help to establish its data requirements. However, it is recognised that without sets of examples familiar to domain experts, the approach advocated in this thesis could lead to the defining of requirements that may later prove to be erroneous or incomplete.

7.3.3 Comparisons with Traditional Approaches

Another approach for improving the credibility of the findings of this research would have been to re-work an existing system developed using traditional analysis tools and techniques. This would have allowed a direct comparison between the conceptual approach adopted in this thesis and a more conventional development strategy. After all, it could be argued that a similar outcome in terms of the quality of the new mentoring sub-system could have been achieved using traditional analysis tools and

techniques. However, more conventional approaches are most often descriptive rather than prescriptive in the nature of their application (Atkins, 1996). Consequently, the quality of data requirements captured using traditional approaches are often influenced by the experience of the analyst who applies them. By comparison, the construction of ORM diagrams is associated with more detailed steps and guidelines and is thus more prescriptive in nature. This suggests that models developed using ORM constructs are less influenced by the experience of the analyst. Indeed, the researcher was able to define a set of data requirements that met all the quality criteria established during the action research component. This was possible even though this was the first system developed by the researcher using ORM constructs. In addition, the ability of ORM to capture many more categories of business rules than traditional modelling tools resulted in a schema that was more complete than could be ordinarily achieved. The verbalisation of business rules also allowed omissions and misconceptions to be identified by domain experts before the generation of the logical schema. All these factors suggest that high quality data models are at least easier to produce by expressing business rules using ORM constructs than is generally the case using traditional approaches.

Unfortunately, it was not possible to perform a direct comparison between traditional modelling techniques and the approach advocated in this thesis to test the argument given above. In re-working an existing system, the researcher could not guarantee that the database design resulting from the suggested approach would not be significantly different from that already in use by the PTE. Since all of the organisation's applications draw upon data from the existing database structure, it was considered unfeasible to modify that structure. Instead, the researcher was limited by

the organisation's management to the development of a new sub-system that would augment the current database structure, rather than necessitate significant changes.

7.4 FUTURE RESEARCH

Possibly the most obvious area of future research would be to extend the number of cases by applying the approach advocated in this thesis in a variety of different problem domains. If the findings documented in Chapter 6 are found to be reproducible, this would then strengthen the hypothesis that has been presented. Since Microsoft has now incorporated Visio Modeller into the Enterprise Architect edition of its .NET product, it is possible that ORM will be used more frequently as a means defining data requirements. If the popularity of this product does escalate over the next few years, the scope for investigating the hypothesis presented would improve correspondingly. Increasing the number of cases may also facilitate quantitative analysis techniques, although it is recognised that quantifying the variables of interest could be problematic. Establishing quality metrics maybe one solution, but the metrics chosen and the values given against each would still be open to interpretation.

This thesis has examined an alternative approach for the specification of data requirements. A natural extension of this approach would be to develop a complementary conceptual process modelling technique to define system processing requirements. Within this thesis, Business Process Modelling (BPM) was used to document these details. However, BPM documents physical implementation details such as the role or person who performs a particular task. The domain is therefore

modelled at a lower level of abstraction than what could be achieved using a conceptual approach. An interesting and related research question would be to determine whether modelling processing requirements at a conceptual level offers any advantage over traditional logical and physical models. Although the researcher encountered no significant problems in validating BPM's with domain experts, a conceptual approach may facilitate a more collaborative and complete definition of the problem domain. Conceptual process modelling constructs could also be applied to more completely specify derivation rules than can currently be achieved using FORML. As documented in section 5.8, FORML can be used to specify simple algorithms such as an invoice total calculation, but more complex rules involving selection and iteration currently require other techniques. Providing a technique that could document such details at the conceptual level, may allow domain experts to more fully participate in their validation.

Possibly the most ambitious area of further research would be to incorporate a conceptual approach for specifying data, processing and non-functional requirements into a complete methodology. The methodology would guide analysts and developers from the initial problem definition to the completion of the project. However, a methodology would also need to define detailed steps with specific tasks and provide tools and techniques with which to complete those tasks. ORM's Conceptual Schema Design Procedure (CSDP) possesses these characteristics, but the approach defines only the data requirements of a system. Thus CSDP is best described as a prescriptive data modelling technique. However, CSDP combined with a prescriptive conceptual technique for defining processing and non-functional requirements could be considered a complete methodology. Establishing a methodology that allows domain

experts to collaborate with analysts and challenge their perceptions could provide a significant tool to address the current high level of IT project failure.

Bibliography

Alavi, M., & Carlson, P. (1992). A review of MIS research and disciplinary development. *Journal of Management Information Systems*, *8*(4), 45-62.

Asymetrix Corporation (1994). *InfoModeler User Manual*. Washington: Asymetrix.

Atkins, C. F. (1996). Prescription or Description: Some Observations on the Conceptual Modelling Process. *Proceedings of Software Engineering: Education and Practice Conference*.

Avison, D., Baskerville, R., & Myers, M. (2001). Controlling Action Research Projects. *Information Technology and People*, *14*(1), 28-45.

Barker, R. (1990). *CASE*Method - Entity Relationships Modelling*. Boston: Addison-Wesley.

Baskerville, R. L. (1999). Investigating Information Systems with Action Research. *Communications of the Association for Information Systems*, *2*(19).

Baskerville, R., & Wood-Harper, A. (1996). A Critical Perspective on Action Research as a Method of Information Systems Research. *Journal of Information Technology*, *11*, 235-246.

Bell, T. E., & Thayer T.A. (1976). Software Requirements: Are They Really a Problem? *Second International Conference on Software Engineering, San Francisco*, pp. 61-68.

Benbasat, I., Goldstein, D. K., & Mead, M. (1987). The Case Research Strategy in Studies of Information Systems. *MIS Quarterly*, *11*(3), 369-386.

Benbasat, I., & Weber, R. (1996). Research Commentary: Rethinking Diversity in Information Systems. *Information Systems Research*, *7*(4), 389-399.

Booch, G.; Rambaugh, J., and Jacobson, I. (1999). *The unified Modelling Language User Guide*. Reading, MA: Addison-Wesley.

Campbell, L. J., & Halpin, T. A. (1994). Abstraction Techniques for Conceptual Schemas. *Proceedings of the 5th Australasian Database Conference, Christchurch*, *16*, 374–388.

Campbell, L. J., Halpin, T. A., & Proper H.A. (1996). Conceptual Schemas with Abstractions – Making flat conceptual schemas more comprehensible. *Data & Knowledge Engineering, 20*(1), 39–85.

Carlson, C. R., Ji, W., & Arora, A. K. (1990). The Nested Entity-Relationship Model. *Proceedings of the Eighth International Conference on Entity-Relationship Approach, Toronto*, pp. 43–57.

Cecez-Kecmanovic, D. (2001). Doing Critical IS Research: The Question of Methodology. In: E. Trauth (Ed.), *Qualitative Research in IS: Issues and Trends*. Hershey: IDEA Group Publishing.

Charmaz, K. (2000). Grounded Theory: Objectivist and Constructivist Methods. In: Denzin, N. & Lincoln, Y. (Eds.), *The Handbook of Qualitative Research*. New York: Sage Publications.

Checkland, P. (1981). *Systems Thinking, Systems Practice*. London: John Wiley.

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

Chua, W. (1986). Radical Developments in Accounting Thought. *The Accounting Review*, *16*(4), 601-632.

Conference Board Survey (2001). *ERP Trends Research Report* (Report No. R-1292-01-RR). New York: ERP Trends.

Cornford, T., & Smithson, S. (1996). *Project Research in Information Systems*. London: Macmillian.

Dalianis, H. (1995). Aggregation, formal specification and natural language generation. *First International Workshop on the Application of Natural Language to Data Bases Versailles*, pp. 39-49.

Date, C. J. (2000). *What, Not How: The Business Rules Approach to Application Development*. Reading, Massachusetts: Addison-Wesley.

Davies, A. M. (1988). A Comparison of Techniques for the Specification of External Behaviour. *Communications of the ACM*, *31*:1098-1115.

DeMarco, T. (1978). *Structured Analysis and System Specification*. New Jersey: Yourdon Press.

DeTroyer, O. and Meersman, R. (1995). A Logic Framework for a Semantics of Object-Oriented Data Modeling. *Proceeding of the 14th International Conference on*

Object Orientation and Entity Relationship Modelling.

Denzin, N. & Lincoln, Y. (Eds.) (2000), *The Handbook of Qualitative Research (2nd Ed.)*. California: Sage Publications.

Downs, E.; Clare, P., & Coe, I. (1988). *Structured Systems Analysis and Design Method*. London: Prentice Hall.

Feather, M. (1987). Language Support for the Specification and Development of Composite Systems. *ACM Transactions on Programming Languages and Systems*, *9*(2), 198-234.

Feldman, P., & Miller, D. (1986). Entity Model Clustering: Structuring a Data Model by Abstractions. *The Computer Journal*, *29*(4), 348-360.

Fellbaum, C. (Ed.) (1998). *WordNet: An Electronic Lexical Database*. Cambridge, Massachusets: MIT Press.

Gage, N. L. (1994). The Scientific Status of Research on Teaching. *Educational Researcher*, *44*(4), 371-383.

Galliers, R. (1992). Choosing Information Systems Research Approaches. In: R. Galliers (Ed.), *Information Systems Research - Issues, Methods and Practical Guidelines*. Oxford: Blackwell Scientific Publications.

Gamut, L. (1991). *Logic, Language, and Meaning*. Chicago: University of Chicago Press.

Gane, C. and Sarson, T. (1979). *Structured Systems Analysis: Tools and Techniques.*

Englewood Cliffs: Prentice Hall.

Glass, R. L. (1997). *Software Runaways*. New Jersey: Prentice Hall.

Glesne, C. and Peshkin, A. (1992). *Becoming Qualitative Researchers*. New York: Longman.

Glinz, M. (1995). An Integrated Formal Model of Scenarios Based on Statecharts. *The Fifth European Software Engineering Conference, Barcelona*, pp. 254-271.

Greenspan, S. J., Mylopoulos, J., & Borgida, A. (1982). Capturing More World Knowledge in the Requirements Specification. *Sixth International Conference on Software Enginering, Tokyo*, pp. 225-234.

Greenspan, S. J., Mylopoulos, J., & Borgida, A. (1986). A Requirements Modelling Language and its Logic. *Information Systems*, *11*(1), 9-23.

Greenspan, S. J.; Mylopoulos, J., and Borgida, A. (1994). On Formal Requirements Modeling Languages: RML Revisited. *Proceedings of the 16th International Conference on Software Engineering*Los Alamitos, pp135-147: IEEE Press.

GUIDE Business Rules Project. (1995) *Defining Business Rules ~ What Are They Really?* Retrieved March 14th, 2003 from http://www.businessrulesgroup.org/first_paper/br01c0.htm.

Halpin, T. (1993). What is an elementary fact? *Proceedings of First NIAM-ISDM Conference,* Utrecht.

Halpin, T. A. (1996, October). Business rules and object role modelling. *Database Programming and Design*, pp. 16-22.

Halpin, T. A. (2001). *Information modeling and relational databases*. San Diego: Academic Press.

Halpin, T. A. (2002). Join Constraints. *Seventh International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design, Toronto*, pp. 121-131

Halpin, T. A., & Bloesch, A. C. (1999). Data modeling in UML and ORM: a comparison. *Journal of Database Management*, *10*(4), 4-13.

Halpin, T. A., & Orlowska, M. E. (1992). Fact-orientated modelling for data analysis. *Journal of Information Systems*, *2*(2), 97-119.

Hirschheim, R. A., Klein, H. K., & Lyytinen, K. (1995). *Information systems development and data modeling : conceptual and philosophical foundations*. Cambridge: Cambridge University Press.

Hirschheim, R., & Newman, M. (1991). Symbolism and Information Systems Development: Myth, Metaphor and Magic. *Information Systems Research*, *2*(1), 29-62.

Holzblatt, K., & Beyer, H. (1993). Making Customer Centred Design Work for Teams. *Communications of the ACM*, *36*(10), 93-103.

Hoppenbrouwers, J., van den Heuvel, W. J., Hoppenbrouwers, S., Weigand, H., & de Troyer, O. (1997) *The grammalizer: A case tool based on textual analysis.* Retrieved March 2nd, 2003, from http://citeseer.nj.nec.com/article/hoppenbrouwers97grammalizer.html.

Hoppenbrouwers, J., van der Vos, B., & Hoppenbrouwers, S. (1997). NL Structures and Conceptual Modelling: Grammalizing for KISS. *Data & Knowledge Engineering*, *23*(1), 79-82.

Hoppenbrouwers, J. J., Hoppenbrouwers, S. J., & Vos, J. v. d. (1996). NL structures and conceptual modelling: The KISS case. In Riet, R.P. van de, Burg, J.F., Vos, A.J. van der (Ed.). *Applications of Natural Language to Information Systems*, pp. 197-209.

Irani, Z., Ezingeard, J., Grieve, R. J., & Race, P. (1999). A Case Study Strategy as Part of an Information Systems Research Methodology: A Critique. *International Journal of Computer Applications*, *12*(5), 190-198.

Jones, C. B. (1990). *Systematic Software using VDM.* (2nd ed.). New York: Prentice Hall.

Kaplan, B., & Duchon, D. (1988). Combining Qualitative and Quantitative Methods in Information Systems Research: A Case Study. *MIS Quarterly*, *12*(4), 571-587.

Kaplan, B. and Maxwell, J. A. (1994). Qualitative Research Methods for Evaluating Computer Information Systems. In: J. Anderson, C. Aydin and S. Jay (Eds.), *Evaluating Health Care Information Systems, Methods and Applications*. Thousand Oaks: Sage.

Klein, H. K. & Hirschheim, R. (1991). Rationality Concepts in Information System Development Methodologies. *Accounting, Management and Information Technology*. *1*(2):157-187.

Klein, H., & Myers, M. (1999). A Set of Principles for Conducting and Evaluating

Interpretive Field Studies in Information Systems. *MIS Quarterly*, *23*(1), 67-94.

Kling, R. (1987). Defining the Boundaries of Computers Across Complex Organisations. In: R. Boland and R. Hirschheim (Eds.), *Critical Issues in Information Systems Research*. London: John Wiley.

Kristen, G. (1994). *Object-Orientation: The Kiss Method : From Information Architecture to Information System*. Boston: Addison-Wesley Longman.

Le Charlier, B., & Flener, P. (1998). Specifications are necessarily informal, or: Some more myths of formal methods. *Journal of Systems and Software*, *40*(3), 275-296.

Leffingwell, D. (1997). Calculating the return on investment from more effective requirements management. *American Programmer, 10*(4), 13-16.

Lewin, K. (1946). Action Research and Minority Problems. *Journal of Social Issues*, *2*(4), 34-46.

Liebscher, P. (1998). Quantity with Quality? Teaching Quantitative and Qualitative Methods in an LIS Master's Program. *Library Trends*, *46*(4), 668-680.

Manna, Z., & Pnueli, A. (1992). *The Temporal Logic of Reactive and Concurrent Systems*. Heidelberg: Springer-Verlag.

Martin, J. and Finkelstein, C. (1981). *Information Engineering (Volumes 1 and 2)*. New Jersey: Prentice Hall.

Martin, P. Y., & Turner, B. A. (1986). Grounded Theory and Organizational Research. *The Journal of Applied Behavioral Science*, *22*(2), 141-157.

Mazza, C., Fairclough, J., & Melton, B. (1994). *Software engineering standards*. New York: Prentice Hall.

Mingers, J. (2001). Combining IS Research Methods: Towards a Pluralist Methodology. *Information Systems Research*, *12*(3), 240-259.

Meyer, B. (1985). On Formalism in Specifications. *IEEE Software*, *2*(1), 6-26.

Mumford, E. (1981). Systems, Objectives, Solutions. *Participative Systems Design: Structure and Method, 1*(1), 5-19.

Mylopoulos, J., Chung, L., & Nixon, B. (1992). Representing and using non functional requirements: A process-orientated approach. *IEEE Transactions on Software Enineering, 18*(6), 483-497.

Mylopoulos, J., Chung, L., & Yu, E. (1999). From Object-Oriented to Goal-Oriented Requirements Analysis. *Communications of the ACM, 42*(1), 31-37.

National Audit Office. (2001). *Why IT Projects Fail*. UK National Audit Review of IT Projects. Retrieved April 25th, 2003 from http://www.nao.gov.uk/intosai/edp/pas/010105ukpaperv2.1.pdf.

New Zealand Qualifications Authority. *The National Qualifications Framework*. Retrieved April 16th, 2004 from http://www.nzqa.govt.nz/framework/explore/index.do.

Nijssen, G. M. & Halpin, T. A. (1989). *Conceptual Schema and Relational Database Design: A fact oriented approach*. Sydney: Prentice-Hall.

OASIG. (1996). *The performance of Information Technology and the role of human and organisational factors*. Sheffield: Institute of Work Psychology, Sheffield University.

Orlikkowski, W., & Baroudi, J. (1991). Studying IT in Organisations: Research Approaches and Assumptions. *Information Systems Research*, *2*(1), 1-28.

Orlikowski, W. (1993). Case Tools as Organisational Change: Investigating Incremental and Radical Changes in Systems Development. *MIS Quarterly*, *17*(3).

Peterson, J. L. (1981). *Petri Net Theory and the Modelling of Systems*. New Jersey: Prentice Hall.

Pinsonneault, A., & Kreamer, K. (1993). Survey Research Methodology in Management Information Systems. *Journal of Management Information Systems*, *10*(2), 75-105.

Plihon, V., Ralyté, J., Benjamen, A., Maiden, N., Sutcliffe, A., Dubois, E., & Heymans, P. (1998). A Reuse-Oriented Approach for the Construction of Scenario Based Methods. *International Software Process Association's Fifth International Conference on Software Process, Chicago*.

Pohl, K. (1993). The Three Dimensions of Requirements Engineering. *Fifth International Conference of Advanced Information Systems Engineering, Paris*, pp. 275-292.

Potter, B., Sinclair, J., & Till, D. (1996). *An Introduction to Formal Specification and Z* (2nd Ed.). London, New York: Prentice Hall.

Potts C., Takahashi, K., & Antňn, A. I. (1994). Inquiry-based Requirements Analysis. *IEEE Software, 11*(2), 21-32.

Rapoport, R. N. (1970). Three Dilemmas in Action Research. *Human Relations, 23*(4), 499-513.

Rolland, C., & Ben Achour, C. (1998). Guiding the Construction of Textual Use Case Specifications. *Data and Knowledge Engineering, 25*(1), 125-150.

Rumbaugh, J., Jacobson, I., & Booch, G. (1999). *The Unified Modeling Language Reference Manual.* Reading, M.A.: Addison-Wesley.

Ryan, K. (1993). The Role of Natural Language in Requirements Engineering. *IEEE International Symposium on Requirements Engineering, San Diego*, pp. 240-242.

Sandy, G. A. (1994). Nice Models, But Where's The Business. *Proceedings of the 5th Australian Conference on Information Systems,* Melbourne.

Sawyer, S. (2001). Analysis by Long Walk: Some Approaches to the Synthhesis of Multiple Sources of Evidence. In: E.M. Trauth (Ed.), *Qualitative Research in IS: Issues and Trends*. Hershey, PA: Idea Group Publishing.

Scott, J. (1998). Organisational Knowledge and the Intranet. *Decision Support Systems Journal, 23*, 3-17.

Simsion, G. (1994). *Data Modelling Essentials: Analysis, Design and Innovation.* Reinhold: Van Nostrand.

Smith, C. (1990). The Case Study: A Useful Research Method for Information Management. *Journal of Information Technology*, *5*, 123-133.

Stake, R. E. (1995). *The Art of Case Study Research*. Thousand Oaks: SAGE.

Statistics New Zealand (2002). *Information Technology Use in New Zealand, 2001.* Retrieved February 13th, 2004 from http://www.stats.govt.nz.

Statistics New Zealand (2003). *Business Statistics Tables*. Retrieved Feb 13th, 2004 from http://www.stats.govt.nz.

Strauss, A. and Corbin, J. (1990). *Basics of Qualitative Research – Grounded Theory Procedures and Techniques*. Newbury Park: Sage Publications.

Strauss, A., & Corbin J. (1994). Grounded Theory Methodology: An Overview. In: N.Denzin and Y. Lincoln (Eds.), *Handbook of Qualitative Research*. Thousand Oaks, California: Sage Publications.

ter Hofstede, A. H., Proper, A. H., & van der Weide, Th. P. (1994). A conceptual language for the description and manipulation of complex information models. *Seventeenth Annual Computer Science Conference, Australia*, *16,* 157-167.

The Standish Group. (1995). *The CHAOS Report*. Dennis, MA: The Standish Group International.

Walsham, G. (1993). *Interpreting Information Systems in Organisations*. Chichester: Wiley.

Walsham, G. (1995). Interpretive Case Studies in IS Research: Nature and Method. *European Journal of Information Systems*, *4*(2), 74-81.

Wasserman, A. (1979). A Specification Method for Interactive Information Systems. *Proceedings of the Specification of Reliable Software Conference, New York*, pp. 68-79.

Weidenhaupt, K., Pohl, K., Jarke, M., & Haumer, P. (1998). Scenario Usage in System Development: A Report on Current Practice. *Third IEEE International Conference on Requirements Engineering, Colorado*, *15*, 34-45.

Whittaker, B. (1999). What Went Wrong? Unsuccessful Information Technology Projects. *Information Management & Computer Security*, *7*(1), 23-30.

Yu, E., Du Bois, P., Dubois, E., & Mylopoulos, J. (1995). From organization models to system requirements: a 'cooperating agents' approach. *Third International Conference on Cooperative Information Systems, Vienna*, pp. 293-312.

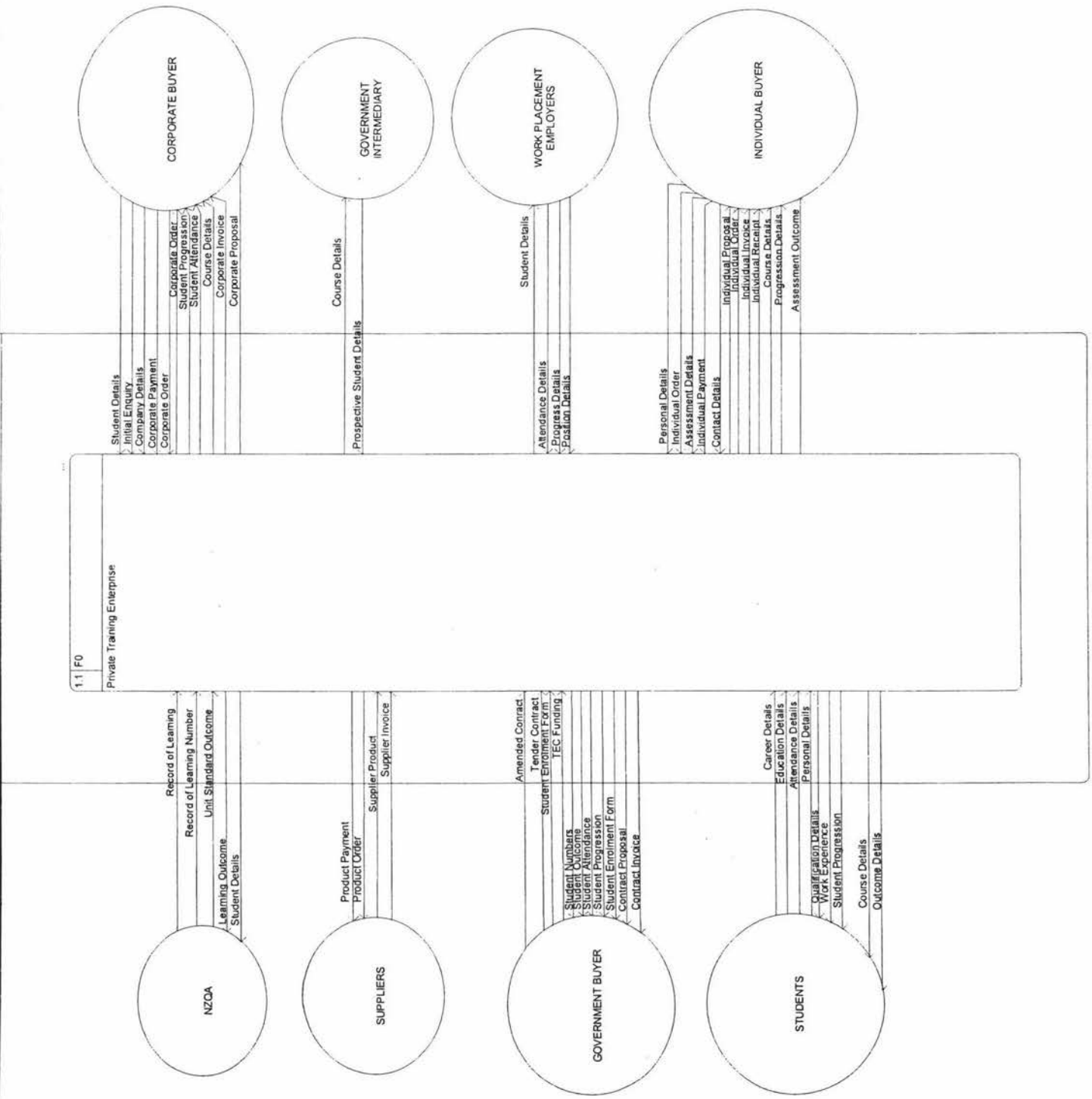
Yue, K. (1987). What Does It Mean to Say that a Specification is Complete? *Fourth International Workshop on Software Specification and Design, Monterey*, pp. 34-41.

Yin, R. (1993). *Applications of Case Study Research*. California: Sage Publications.

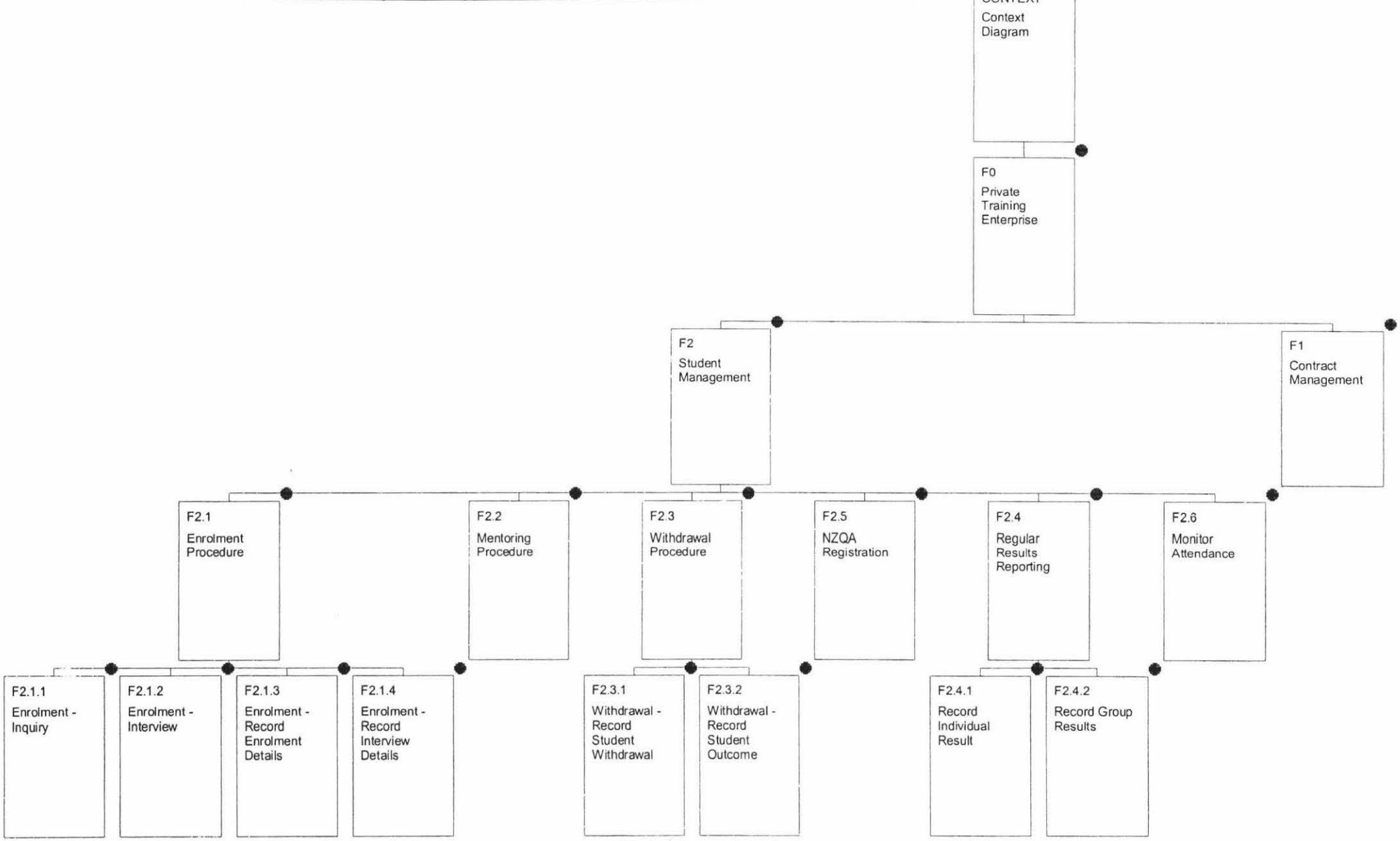
Yin, R. (1994). *Case Study Research, Design and Methods (2nd Ed.).* California: Sage Publications.

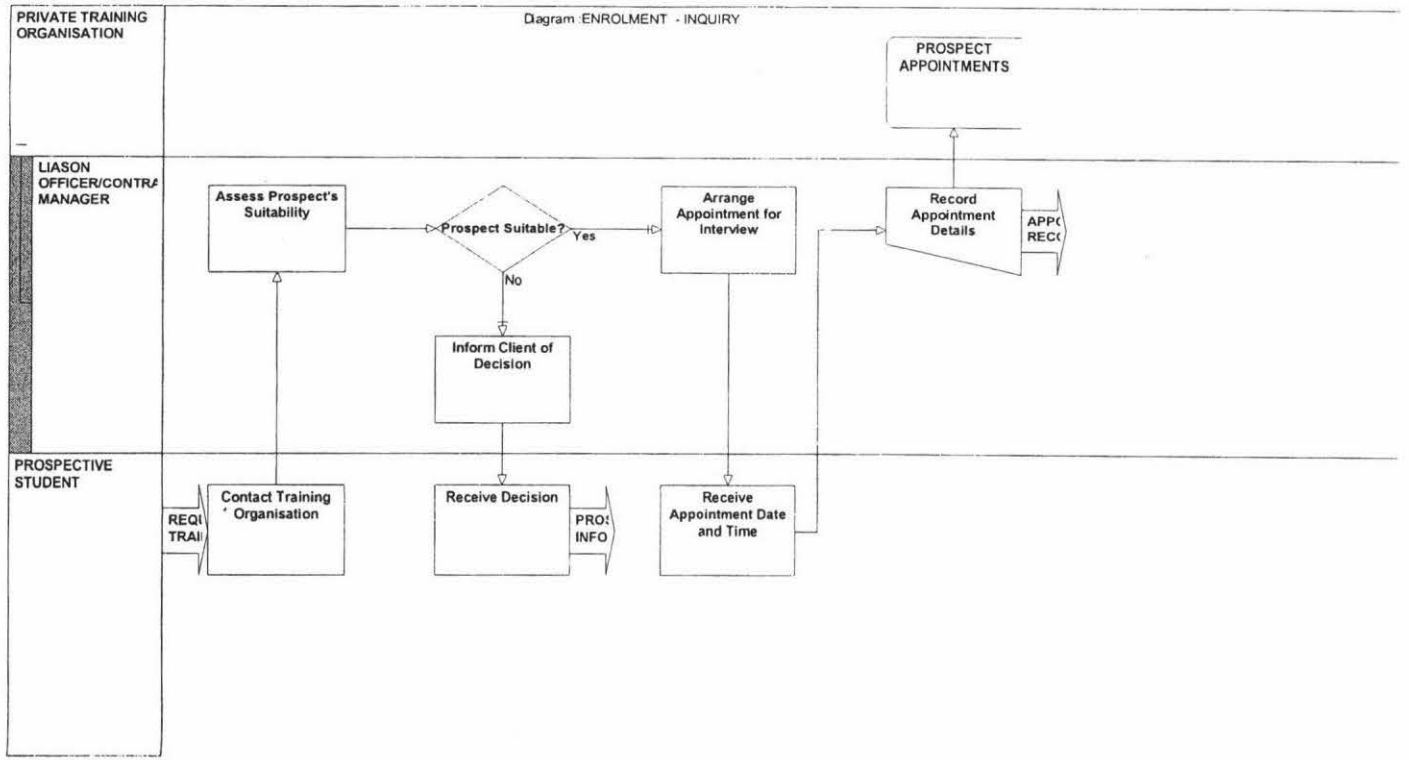
Vitalari, N. P. (1992). Structuring the Requirements Analysis Process for Information Systems: A Proposition Viewpoint. In: W. Cotterman and J. Senn (Eds.), *Challenges and Strategies for Research in Systems Development*. London: John Wiley.

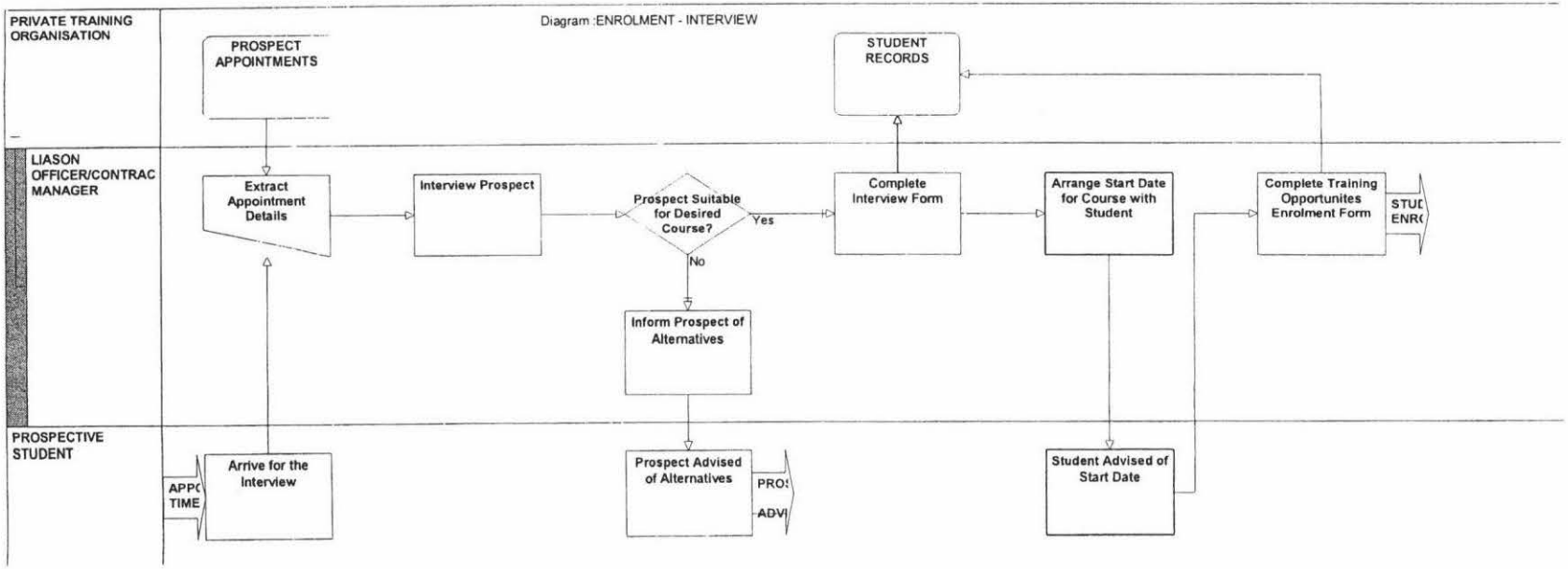
Appendix One – Context Diagram for Private Training Organisation



Appendix 2 – Function Hierarchy Diagram and Process Models for Student Management at the Private Training Enterprise


CONTEXT
Context Diagram
F0
Private Training Enterprise
F2
Student Management
F1
Contract Management
F2.1
Enrolment Procedure
F2.2
Mentoring Procedure
F2.3
Withdrawal Procedure
F2.5
NZQA Registration
F2.4
Regular Results Reporting
F2.6
Monitor Attendance
F2.1.1
Enrolment - Inquiry
F2.1.2
Enrolment - Interview
F2.1.3
Enrolment - Record Enrolment Details
F2.1.4
Enrolment - Record Interview Details
F2.3.1
Withdrawal - Record Student Withdrawal
F2.3.2
Withdrawal - Record Student Outcome
F2.4.1
Record Individual Result
F2.4.2
Record Group Results


PRIVATE TRAINING ORGANISATION
Diagram :ENROLMENT - INQUIRY
PROSPECT APPOINTMENTS
LIASON OFFICER/CONTRA MANAGER
Assess Prospect's Suitability
Prospect Suitable?
Yes
No
Arrange Appointment for Interview
Record Appointment Details
APPC REC
Inform Client of Decision
PROSPECTIVE STUDENT
REQU TRAI
Contact Training ' Organisation
Receive Decision
PROS INFO
Receive Appointment Date and Time


PRIVATE TRAINING ORGANISATION
Diagram :ENROLMENT - INTERVIEW
PROSPECT APPOINTMENTS
STUDENT RECORDS
LIASON OFFICER/CONTRAC MANAGER
Extract Appointment Details
Interview Prospect
Prospect Suitable for Desired Course?
Yes
No
Complete Interview Form
Arrange Start Date for Course with Student
Complete Training Opportunites Enrolment Form
STUI ENR(
Inform Prospect of Alternatives
PROSPECTIVE STUDENT
APP(TIME
Arrive for the Interview
Prospect Advised of Alternatives
PRO!
ADVI
Student Advised of Start Date

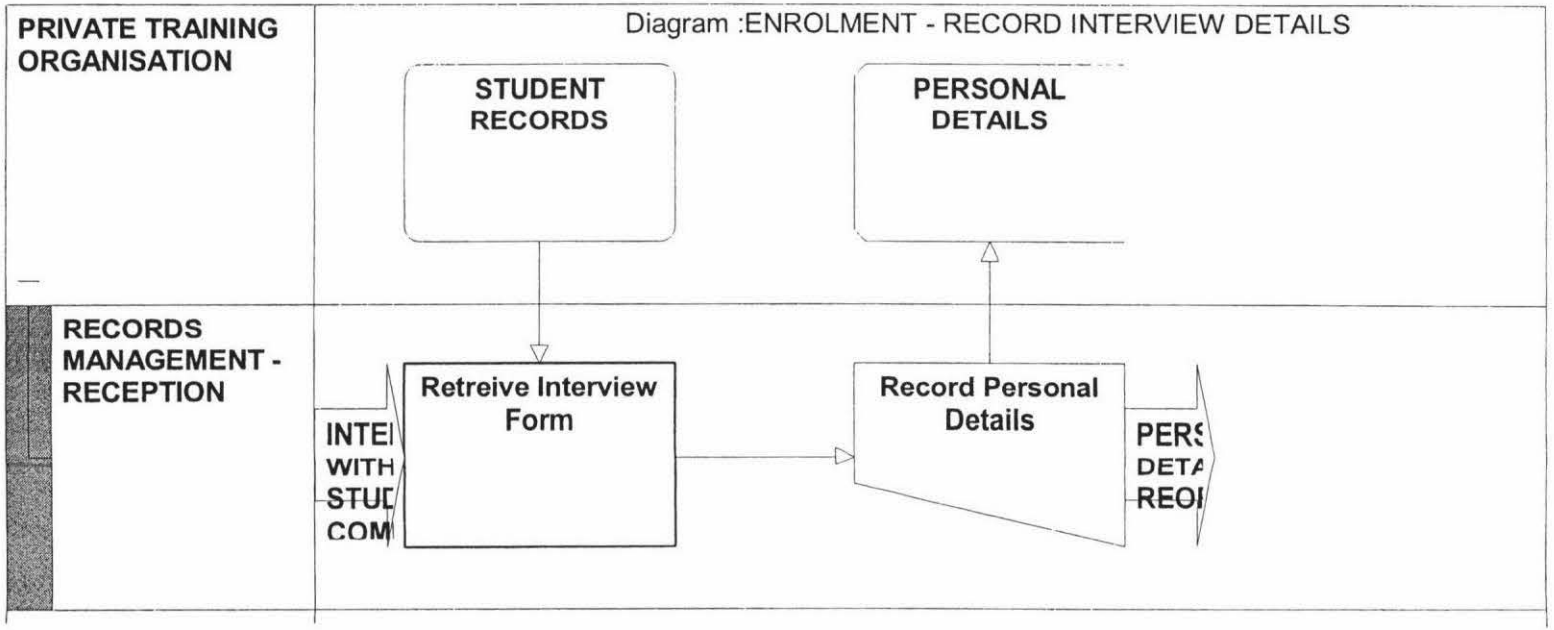
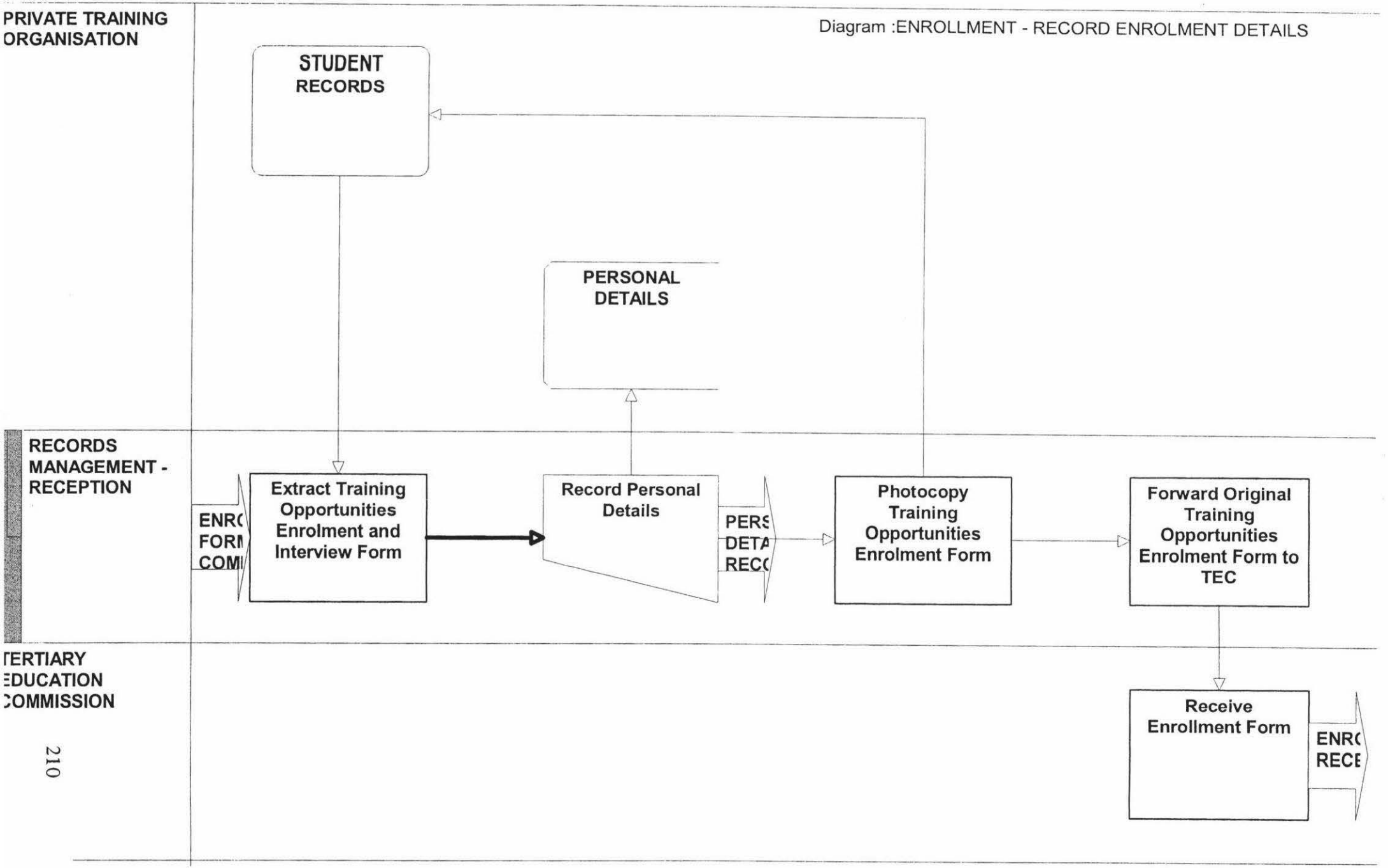
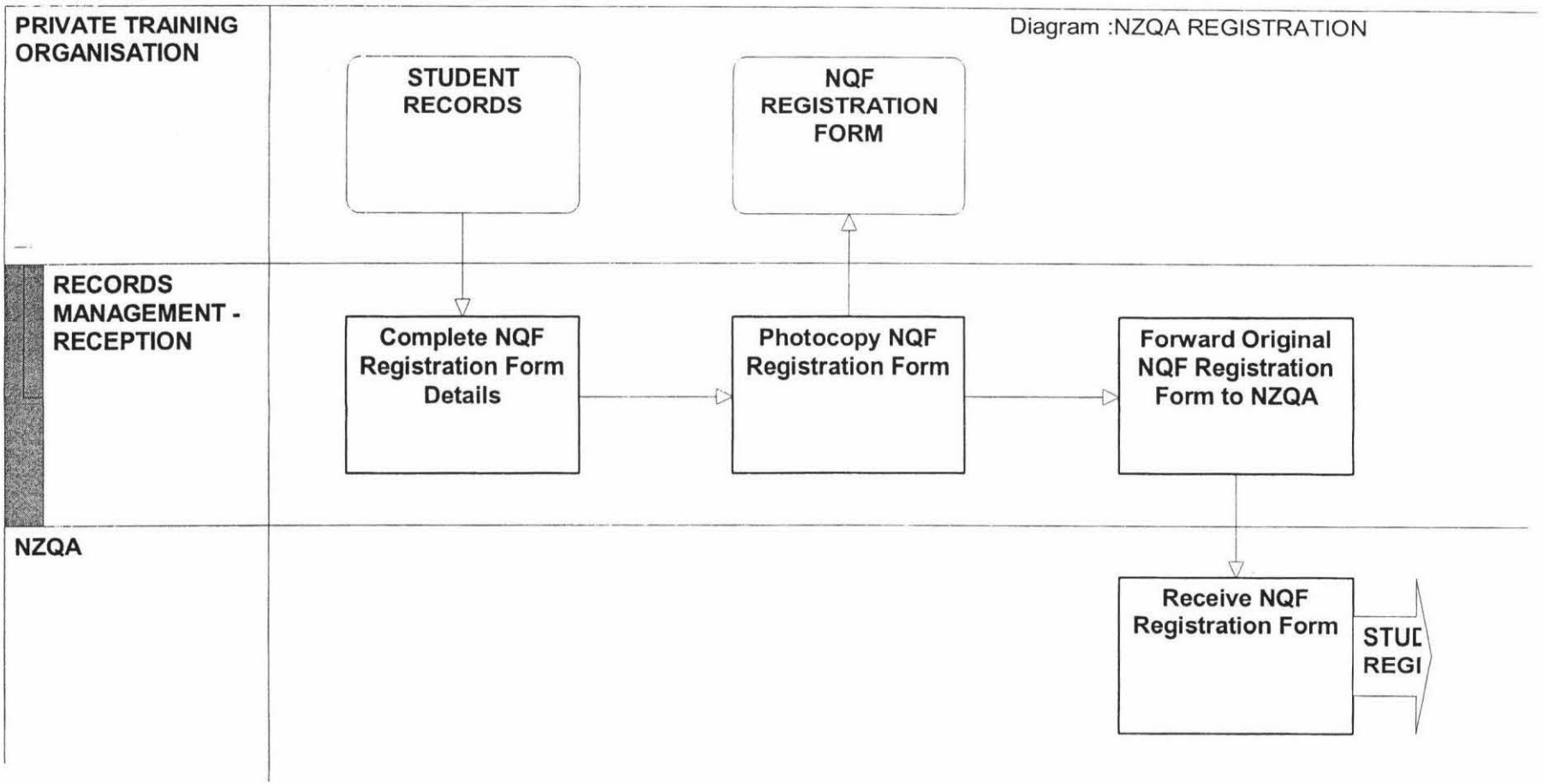
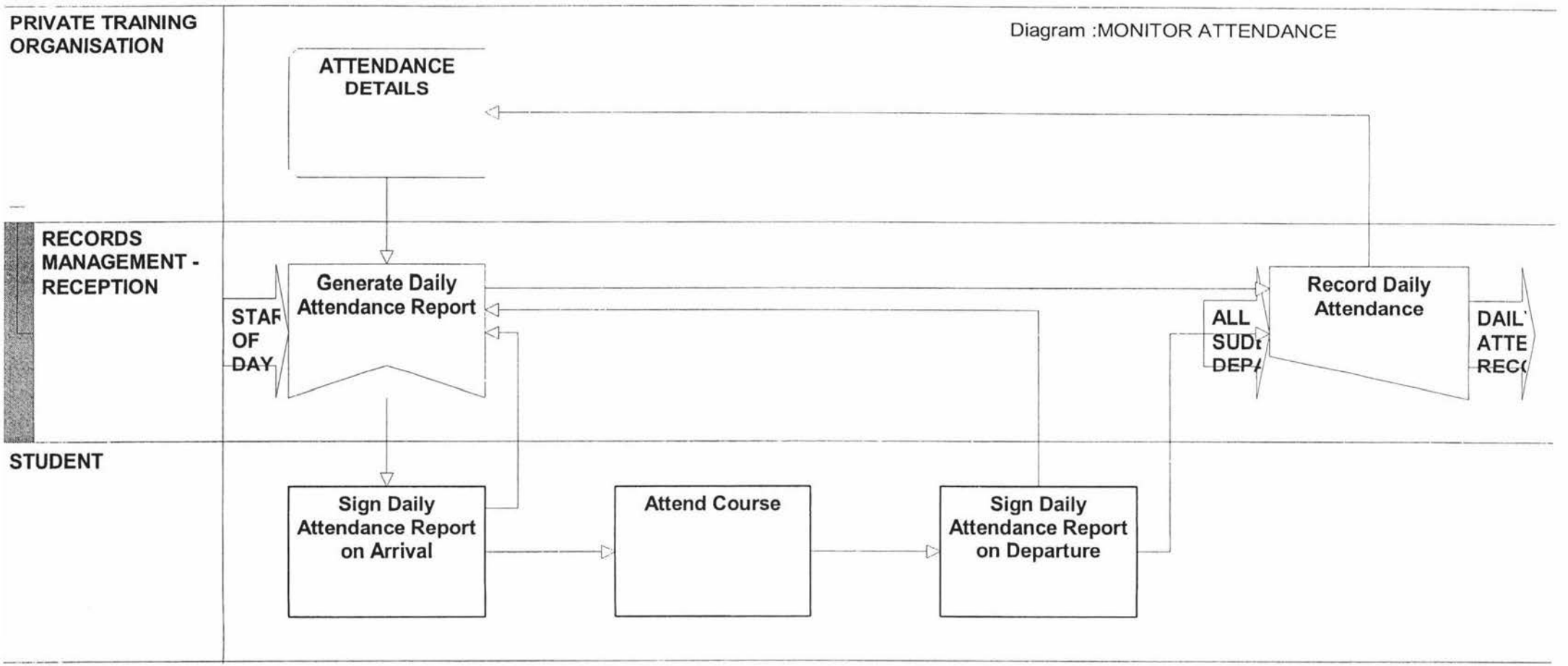
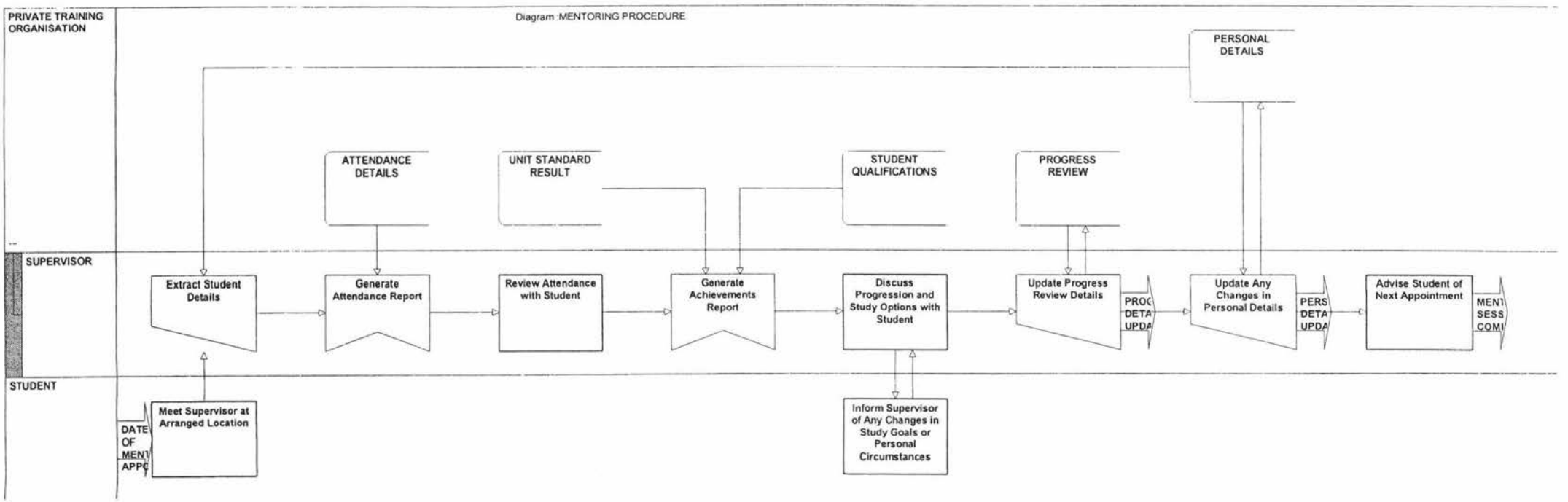

PRIVATE TRAINING ORGANISATION
Diagram :ENROLMENT - RECORD INTERVIEW DETAILS
STUDENT RECORDS
PERSONAL DETAILS
RECORDS MANAGEMENT - RECEPTION
INTE
WITH
STUD
COM
Retreive Interview Form
Record Personal Details
PERS
DETA
REOI

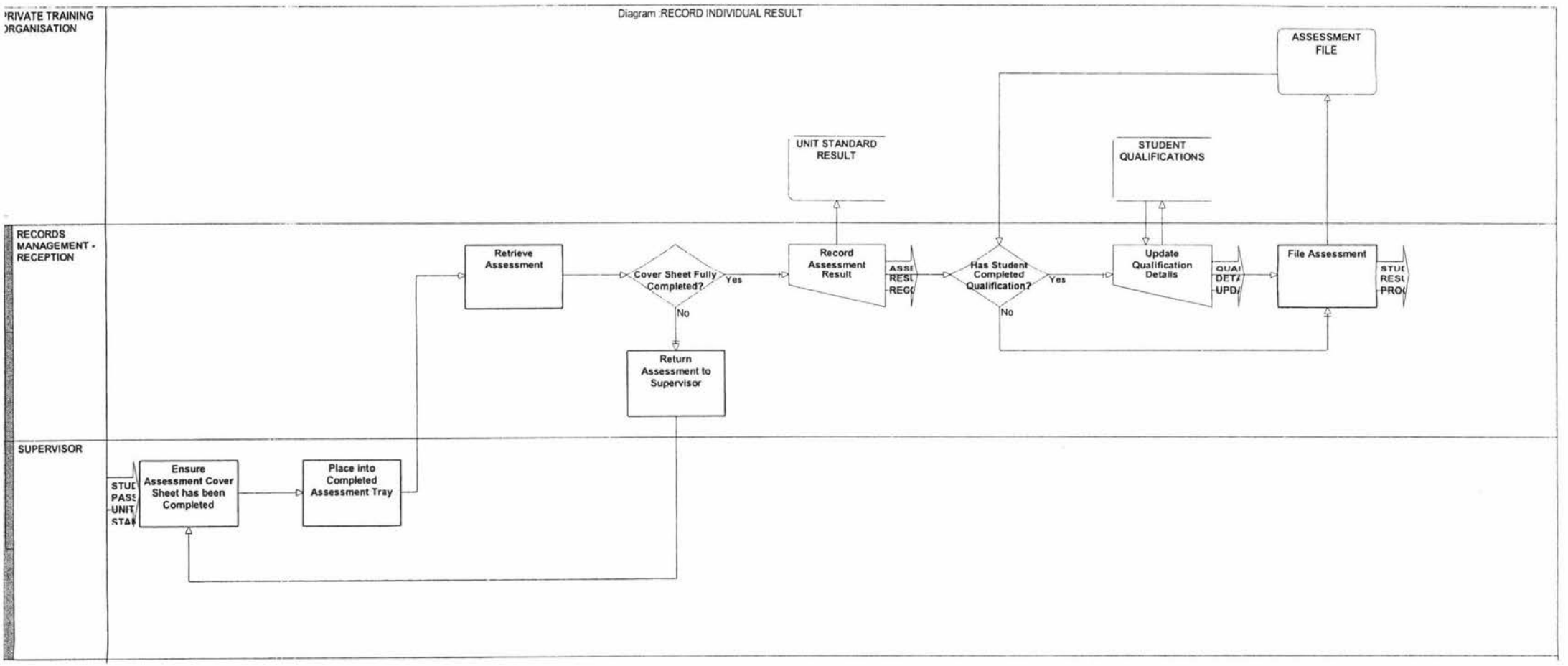
Diagram :ENROLLMENT - RECORD ENROLMENT DETAILS

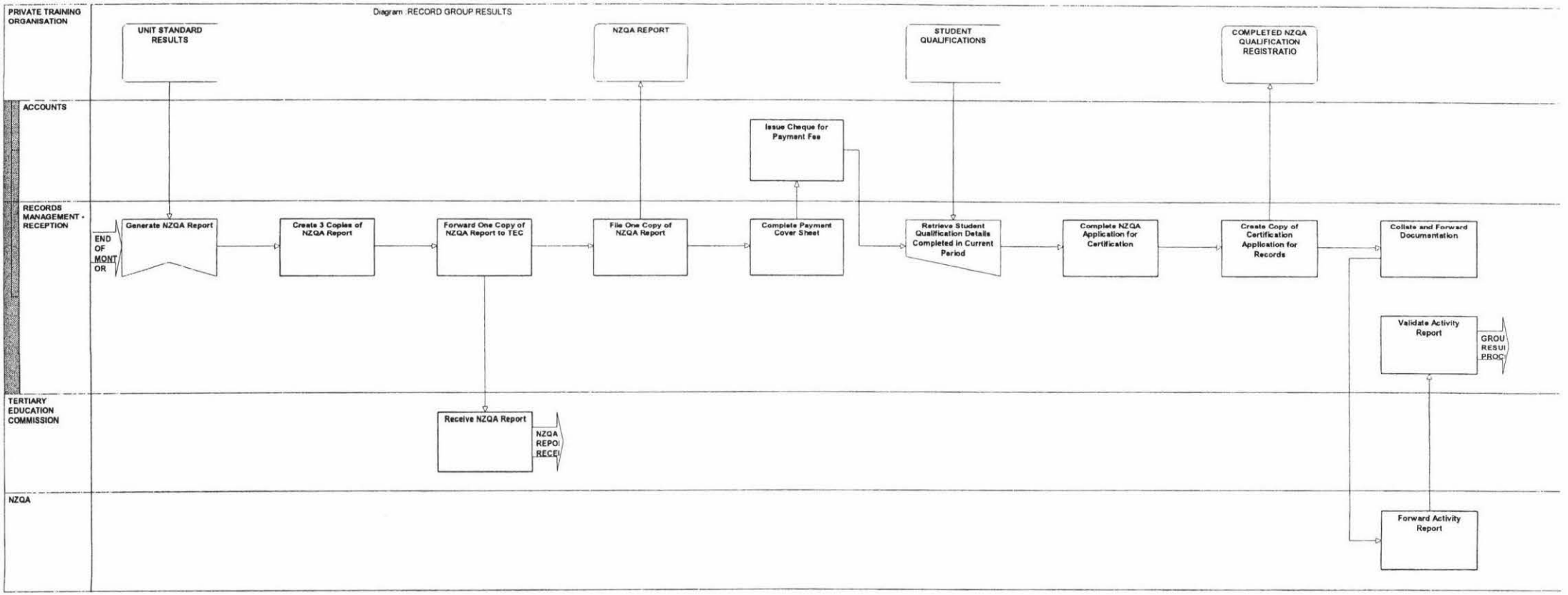


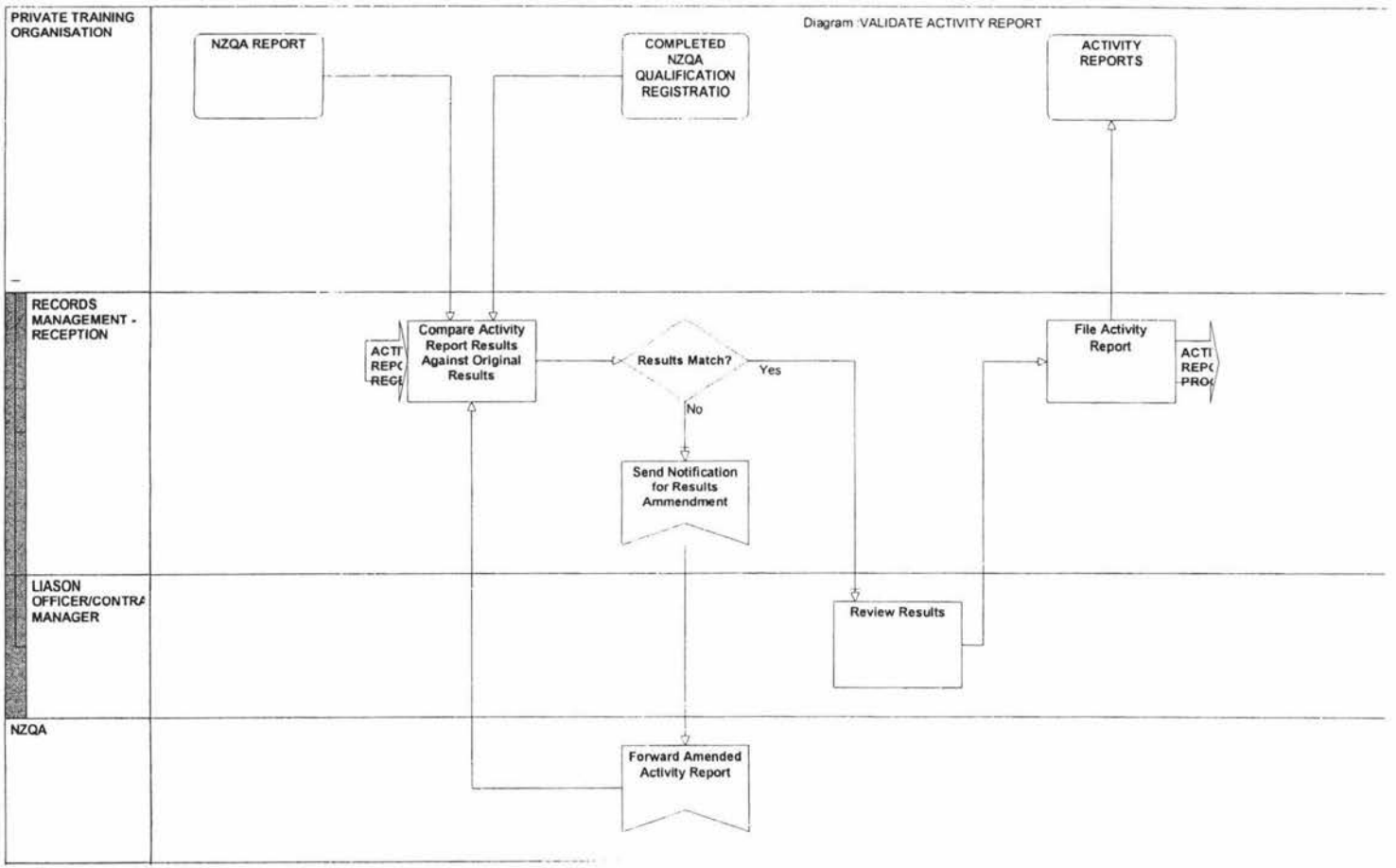

PRIVATE TRAINING ORGANISATION
Diagram :NZQA REGISTRATION
STUDENT RECORDS
NQF REGISTRATION FORM
RECORDS MANAGEMENT - RECEPTION
Complete NQF Registration Form Details
Photocopy NQF Registration Form
Forward Original NQF Registration Form to NZQA
NZQA
Receive NQF Registration Form
STUI
REGI


PRIVATE TRAINING ORGANISATION
Diagram :MONITOR ATTENDANCE
ATTENDANCE DETAILS
RECORDS MANAGEMENT - RECEPTION
STAF
OF
DAY
Generate Daily Attendance Report
ALL
SUD
DEP
Record Daily Attendance
DAIL
ATTE
REC
STUDENT
Sign Daily Attendance Report on Arrival
Attend Course
Sign Daily Attendance Report on Departure


PRIVATE TRAINING ORGANISATION
Diagram :MENTORING PROCEDURE
PERSONAL DETAILS
ATTENDANCE DETAILS
UNIT STANDARD RESULT
STUDENT QUALIFICATIONS
PROGRESS REVIEW
SUPERVISOR
Extract Student Details
Generate Attendance Report
Review Attendance with Student
Generate Achievements Report
Discuss Progression and Study Options with Student
Update Progress Review Details
PROC DETA UPDA
Update Any Changes in Personal Details
PERS DETA UPDA
Advise Student of Next Appointment
MENT SESS COMI
STUDENT
DATE OF MENT APPC
Meet Supervisor at Arranged Location
Inform Supervisor of Any Changes in Study Goals or Personal Circumstances


PRIVATE TRAINING ORGANISATION
Diagram :RECORD INDIVIDUAL RESULT
ASSESSMENT FILE
UNIT STANDARD RESULT
STUDENT QUALIFICATIONS
RECORDS MANAGEMENT - RECEPTION
Retrieve Assessment
Cover Sheet Fully Completed?
Yes
No
Record Assessment Result
Has Student Completed Qualification?
Yes
No
Update Qualification Details
File Assessment
Return Assessment to Supervisor
SUPERVISOR
Ensure Assessment Cover Sheet has been Completed
Place into Completed Assessment Tray


PRIVATE TRAINING ORGANISATION
Diagram :RECORD GROUP RESULTS
UNIT STANDARD RESULTS
NZQA REPORT
STUDENT QUALIFICATIONS
COMPLETED NZQA QUALIFICATION REGISTRATIO
ACCOUNTS
Issue Cheque for Payment Fee
RECORDS MANAGEMENT - RECEPTION
END OF MONTH OR
Generate NZQA Report
Create 3 Copies of NZQA Report
Forward One Copy of NZQA Report to TEC
File One Copy of NZQA Report
Complete Payment Cover Sheet
Retrieve Student Qualification Details Completed in Current Period
Complete NZQA Application for Certification
Create Copy of Certification Application for Records
Collate and Forward Documentation
Validate Activity Report
GROU RESUI PROC
TERTIARY EDUCATION COMMISSION
Receive NZQA Report
NZQA REPO RECE
NZQA
Forward Activity Report


PRIVATE TRAINING ORGANISATION
Diagram :VALIDATE ACTIVITY REPORT
NZQA REPORT
COMPLETED NZQA QUALIFICATION REGISTRATIO
ACTIVITY REPORTS
RECORDS MANAGEMENT - RECEPTION
Compare Activity Report Results Against Original Results
Results Match?
Yes
No
File Activity Report
Send Notification for Results Ammendment
LIASON OFFICER/CONTRA MANAGER
Review Results
NZQA
Forward Amended Activity Report

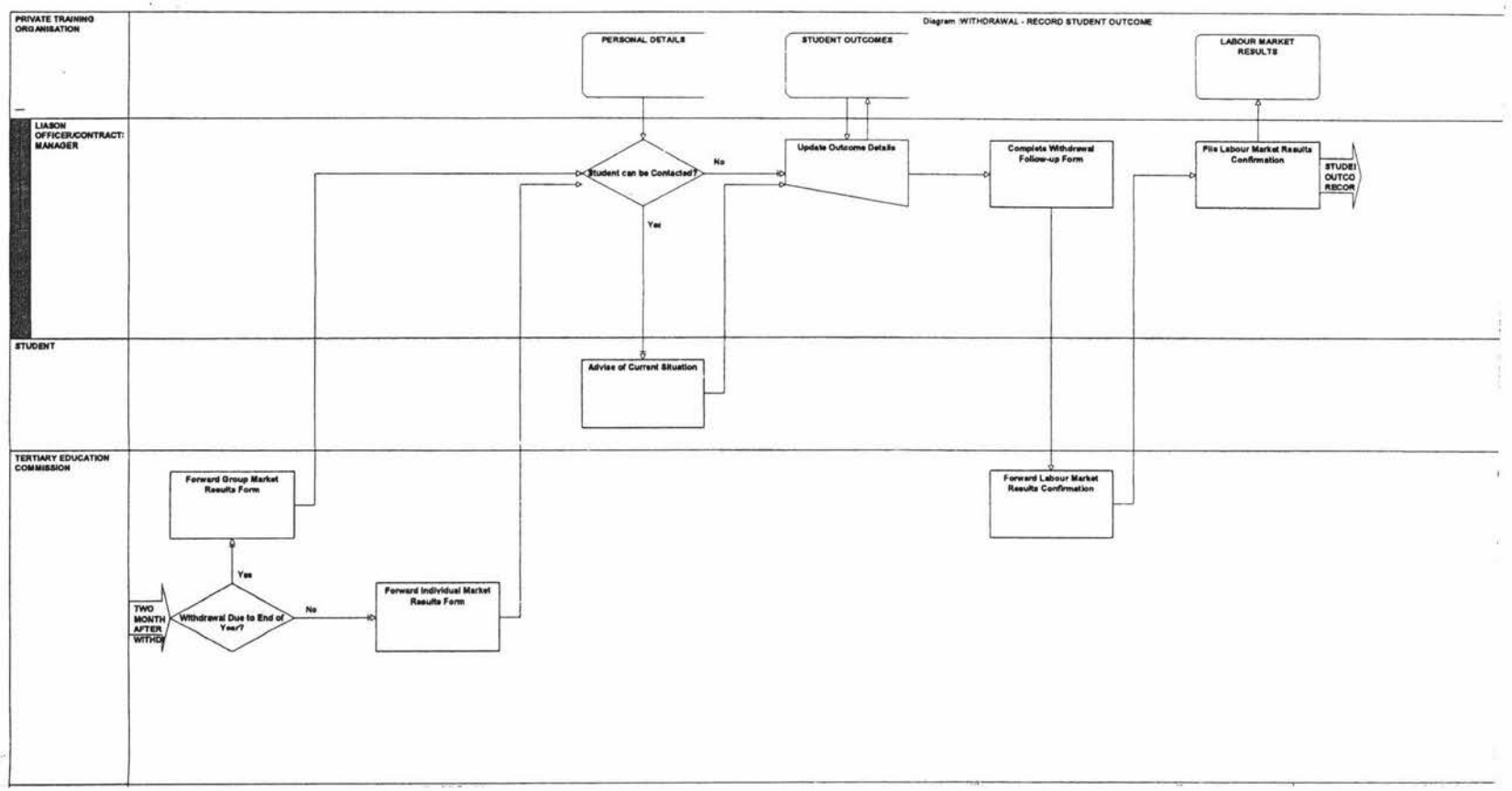


Diagram :WITHDRAWAL - RECORD STUDENT OUTCOME
PRIVATE TRAINING ORGANISATION
PERSONAL DETAILS
STUDENT OUTCOMES
LABOUR MARKET RESULTS
LIASON OFFICER/CONTRACT MANAGER
Student can be Contacted?
No
Yes
Update Outcome Details
Complete Withdrawal Follow-up Form
File Labour Market Results Confirmation
STUDEI OUTCO RECOR
STUDENT
Advise of Current Situation
TERTIARY EDUCATION COMMISSION
Forward Group Market Results Form
Yes
TWO MONTH AFTER WITHDI
Withdrawal Due to End of Year?
No
Forward Individual Market Results Form
Forward Labour Market Results Confirmation

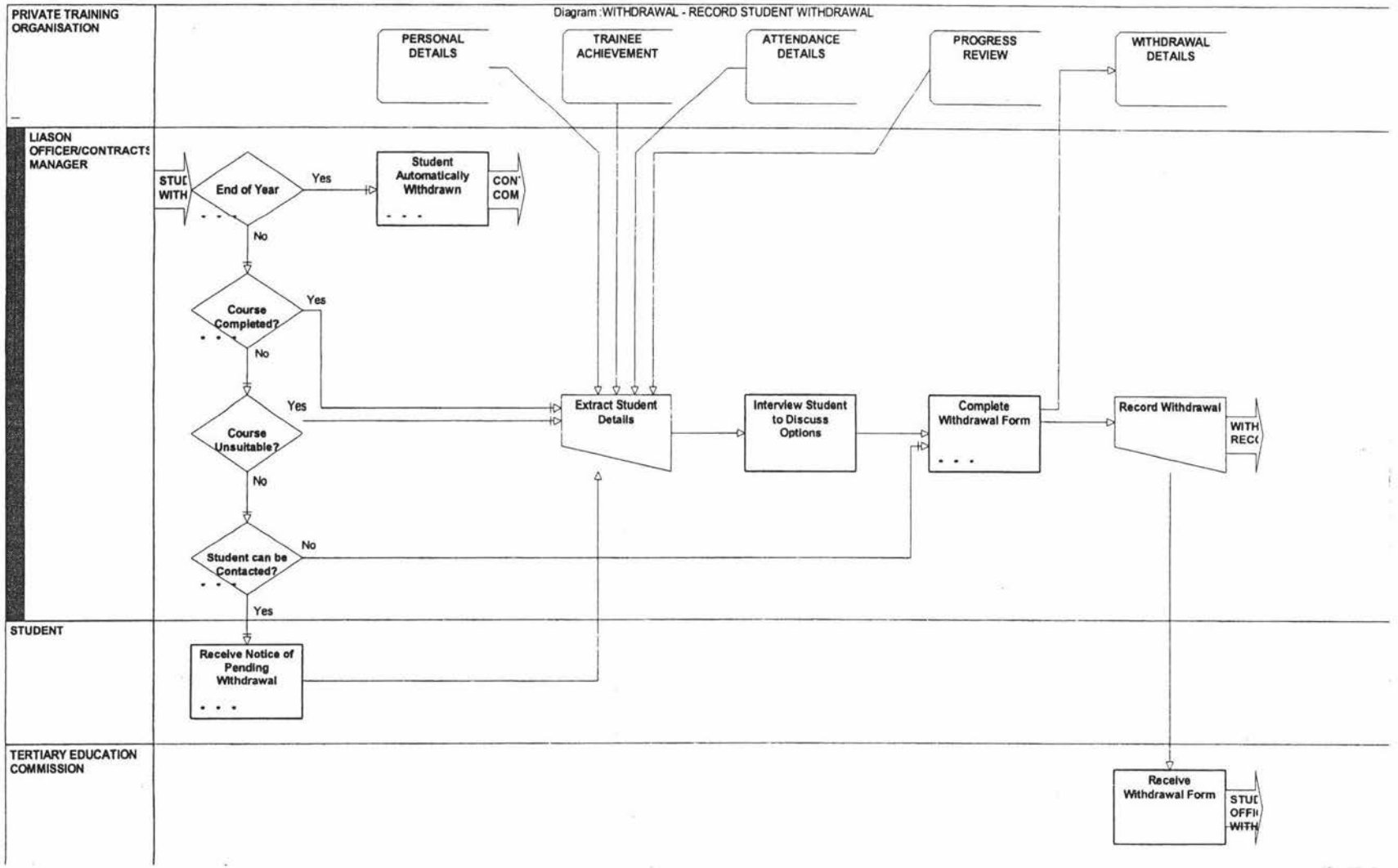


Diagram :WITHDRAWAL - RECORD STUDENT WITHDRAWAL
PRIVATE TRAINING ORGANISATION
PERSONAL DETAILS
TRAINEE ACHIEVEMENT
ATTENDANCE DETAILS
PROGRESS REVIEW
WITHDRAWAL DETAILS
LIASON OFFICER/CONTRACTS MANAGER
STUI WITH
End of Year
Yes
No
Student Automatically Withdrawn
CON COM
Course Completed?
Yes
No
Course Unsuitable?
Yes
No
Student can be Contacted?
No
Yes
Extract Student Details
Interview Student to Discuss Options
Complete Withdrawal Form
Record Withdrawal
WITH REC
STUDENT
Receive Notice of Pending Withdrawal
TERTIARY EDUCATION COMMISSION
Receive Withdrawal Form
STUI OFFI WITH

Appendix 3 – Prototype Student Supervisor Mentoring Report

SUMMARY OF CREDITS ACHIEVED TOWARDS THE
National Certificate in Business Administration and Computing Level 9

Date: DD/MM/YYYY Student: xxxxxxxxxxxxxxx xxxxxxxxxxxxxxx (9999)

CERTIFICATE SUMMARY

Compulsory Requirements	Minimum Required	Achieved	Status
Subfield Requirements: *(only printed if any exist)*			
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx *(Repeated for each required subfield - lists subfield name)*	99	99	Completed *(or blank)*
Domain Requirements:			
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx *(Repeated for each required domain – lists domain name)*	99	99	Completed *(or blank)*
Unit Standard Requirements: (only printed if any exist)			
99999 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx *(Repeated for each required unit standard – lists reference number and title of unit standard)*	99	99	Completed *(or blank)*
Total Credits Required	99	99	Completed *(or blank)*
Total Credits @ Level 9 required *(Repeated for each required level)*	99	99	Completed *(or blank)*

CERTIFICATE (NOT YET) ACHIEVED

Student Achievements and Options

Unit Standards within Required Subfields *(This section only printed when subfield requirements exist for qualification)*

Subfield xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx *(subfield name)*
Domain xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx *(domain name – within subfield)*

Unit No	Unit Standard Title	Level	Credits	Status
99999	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx *(Repeated for each unit standard within domain)*	9	99	Completed *(or blank)*
Total Credits Achieved in this Domain			99	
Total Credits Available in this Domain			99	

(The above section is repeated for each domain within the required subfield)

Unit Standards within Optional Subfields *(This section only printed when optional subfield exist for qualification)*

Subfield xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx *(subfield name)*
Domain xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx *(domain name – within subfield)*

Unit No	Unit Standard Title	Level	Credits	Status
99999	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	9	99	Completed *(or blank)*
(Repeated for each unit standard within domain)				
Total Credits Achieved in this Domain			99	
Total Credits Available in this Domain			99	

(The above section is repeated for each domain within the optional subfield)

Domain Requirements and Options xxxxxxxxxxxxxxxxxx *(name of required or optional domain)*

Unit No	Unit Standard Title	Level	Credits	Status
99999	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	9	99	Completed *(or blank)*
(Repeated for each unit standard within domain)				
Total Credits Achieved for xxxxxxxxxxxxxxxxxx *(domain name)*			99	
Total Credits Required (if any)			99	

(The above section is repeated for each required and/or optional domain)

Unit Standard Options and Achievements

Unit No	Unit Standard Title	Level	Credits	Status
99999	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	9	99	Completed *(or blank)*
(Repeated for each required and/or optional unit standard)				
Total Credits for these Unit Standards			99	

END OF PROGRESS REPORT

Appendix 4
Glossary of Objects (Of Type Entity)

Object	Definition
Achievement	Achievement is an individual student achievement i.e. the particular unit standard that has been successfully completed.
Assessment Group	Assessment Group is a group of Unit Standards achievements for a particular Student. Assessment Groups are used to match against a set of Qualification requirements to determine whether they have been met.
Completed Qualification	Completed Qualification is a Qualification achieved by a student prior to Enrolment onto a McGirr Training Programme.
Unit Standard	Unit Standard is a nationally registered, coherent set of learning outcomes and associated performance criteria, together with technical and management information that supports delivery and assessment. All unit standards are registered on the National Qualifications Framework, assigned a level and a credit value, and may contribute to the award of a National Certificate or Diploma.
Date	Date is the day, month and year specified in the fact that something has occurred.
Domain	Domain is an area of learning within a Subfield on the National Qualifications Framework, as defined in the National Qualifications Framework classification system.
Domain Requirement	Domain Requirement is a requirement specified in an Elective of a Qualification for a certain number of Unit Standard credits from a Domain.
Elective	Elective is a specified set of requirements associated with a particular Qualification. Each Elective specifies the Subfield or Domain from which Unit Standards may (or must) be achieved to complete that Elective. The Elective may also specify a minimum Level and/or Credit Value that Unit Standards must meet to achieve the Elective. The Elective may also specify a particular Unit Standard(s) that may (or must) be achieved. The Elective can be compulsory or optional. An Elective may also be called a 'Strand' in some Qualification specifications.

Field	Field is a broad area of learning on the National Qualifications Framework, as defined in the National Qualifications Framework classification system.
Programme	Programme is a programme of study undertaken by a student at McGirr Training. A programme involves the completion of Unit Standards that may be credited towards a National Certificate or Diploma.
Completed Qualification	Completed Qualification is a Qualification achieved by a student prior to Enrolment onto a McGirr Training Programme.
Student	Student is a learner who is enrolled in a Programme of study at McGirr Training.
Subfield	Subfield is an area of learning within a particular Field on the National Qualifications Framework.
Subfield Requirement	Subfield Requirement is a requirement specified in an Elective of a Qualification for a certain number of Unit Standard credits from a Subfield.

Glossary of Objects (Of Type Value)

Address	Address is the street address of a Student.
Assessment Group Name	Assessment Group Name is a label for an Assessment Group.
City	City is the City in which a Student lives.
Compulsory Status	Compulsory Status is an indication of whether or not a Unit Standard Requirement must be completed by a Student.
Credit Value	Credit Value is a numerical value assigned to a Unit Standard on the National Qualifications Framework that represents the estimated time in hours needed for a typical learner to demonstrate that all specified outcomes have been met.
Domain Name	Domain Name is the name given to a particular Domain as defined in the National Qualifications Framework classification system.
Elective Name	Elective Name is a name given to an Elective of a particular Qualification as specified on the National Qualification Framework's classification scheme.
Ethnicity	Ethnicity is the racial affiliation of a Student.
Field Name	Field Name is the name given to a particular Field as defined in the National Qualifications Framework classification system.
Forename	Forename is the first name of a Student.
Highest Qualification	Highest Qualification is the highest educational achievement of a Student.
Home Telephone	Home Telephone is a Student's home telephone number.
Last School	Last School is the name of the last school attended by a Student.
Level	Level is a descriptor that indicates the complexity of learning associated with each Qualification. There are ten levels involved in a Qualification - 1 is the least complex and 10 the most. They do not equate to 'years spent learning' but reflect the content of the Qualification.
Maximum Credit Exemption	Maximum Credit Exemption is the maximum number of credits that may be granted towards a qualification on the basis of credits already earned in another qualification. Credits have to be recognised before they are transferred.

Maximum Credit Total	Maximum Credit Total is the maximum total credits across all Levels allowed from a particular Qualification's Elective.
Maximum Credit Value	Maximum Credit Value is the minimum number of Unit Standard credits allowed for this Elective at this Level. This detail is often shown on the first page of each qualification word document on the NZQA website. It is often shown together with the minimum credit value for that elective at a particular level i.e. a credit value range. Maximum Credit Value is the upper limit of that range.
Maximum Paper Exemption	Maximum Paper Exemption is the maximum number of National Diploma papers allowed to be used towards meeting the requirements of a qualification. Diploma papers have to be recognised before they are transferred to a qualification on the NQF.
Middle Name	Middle Name is the middle name of a Student.
Minimum Credit Total	Minimum Credit Total is the minimum total credits across all Levels allowed from a particular Qualification's Elective.
Minimum Credit Value	Minimum Credit Value is the minimum number of Unit Standard credits allowed for this Elective at this Level. This detail is often shown on the first page of each qualification word document on the NZQA website. It is often shown together with the maximum credit value for that elective at a particular level i.e. a credit value range. Minimum credit value is the lower limit of that range.
Minimum Group Value	Minimum Group Value is the minimum number of groups from an Elective that students must select in order to meet the requirements for a Qualification. This requirement only effects certain qualifications at level 4 and above (e.g. the N.C. in Business - Level 4).
NQF Reference	NQF Reference is the unique reference number assigned to a Qualification as defined in the National Qualifications Framework (NQF) classification system.
NZQA Reference	NZQA Reference is the unique reference number assigned to a Student who has registered with the New Zealand Qualification Authority (NZQA).

Programme Name	Programme Name is the name given to an area of study at McGirr Training.
Purpose	Purpose is a statement issued by the NZQA as to the intention of a Qualification. Details include an exit profile of a typical student who has completed the Qualification.
Sex	Sex is the gender of the Student.
Subfield Name	Subfield Name is the name given to a particular Subfield as defined in the National Qualifications Framework classification system.
Suburb	Suburb is the suburb in which a Student lives.
Surname	Surname is the last name of the Student.
Tertiary Experience	Tertiary Experience is an indication of whether or not a Student has studied within a post compulsory educational environment e.g. polytechnic, university or another P.T.E
Title	Title is a name assigned to a Qualification as defined in the National Qualifications Framework (NQF) classification system.
Unit Title	Unit Title is the title given to a particular Unit Standard as defined in the National Qualifications Framework classification system.
Work Telephone	Work Telephone is a Student's work telephone number.

Appendix 5 – Business Rules Captured in First Case Study

Achievement is issued an application for accreditation on Date
Achievement is within Assessment Group
Achievement received accreditation on Date

Assessment Group has Assessment Group Name
Assessment Group meets Domain Requirement
Assessment Group meets Subfield Requirement
Assessment Group meets Unit Standard Requirement

Completed Qualification is within Assessment Group
Completed Qualification was accredated on Date
Completed Unit Standard is within Assessment Group
Completed Unit Standard was accredated on Date
Domain contains Unit Standard

Domain has Domain Name
Domain is within Subfield
Domain Requirement has Credit Value
Domain Requirement has Level

Elective has Elective Name
Elective has Maximum Credit Total *
Elective has Maximum Credit Value at Level
Elective has Minimum Credit Total *
Elective has Minimum Credit Value at Level
Elective requires Domain
Elective requires Subfield
Elective requires Unit Standard

Field has Field Name

Programme ends on Date
Programme has Programme Name
Programme starts on Date

Qualification contains Elective
Qualification expires on Date
Qualification has Credit Value
Qualification has Level
Qualification has Maximum Credit Exemption
Qualification has Maximum Paper Exemption
Qualification has Minimum Group Requirement
Qualification has NQF Reference
Qualification has Purpose
Qualification has Title
Qualification is registered on Date
Qualification is within Domain
Qualification is within Field
Qualification is within Subfield

Student completes Unit Standard
Student enrols on Programme on Date
Student has Address
Student has Ethnicity
Student has Final School Year
Student has Forename
Student has Highest Qualification
Student has HomeTelephone
Student has Last School
Student has Middle Name
Student has NZQA Reference
Student has previously achieved Qualification
Student has previously achieved Unit Standard
Student has Surname
Student has Tertiary Experience
Student has Work Telephone
Student is of Sex
Student lives in City
Student lives in Suburb
Student was born on Date
Student withdraws from Programme on Date

Subfield has Subfield Name
Subfield is within Field
Subfield Requirement has Credit Value
Subfield Requirement has Level

Unit Standard expires on Date
Unit Standard has Credit Value
Unit Standard has Level
Unit Standard has Unit Title
Unit Standard Requirement has Compulsory Status

* Derived Value

Appendix 6 – Supervisor Mentoring Reports

SUMMARY OF CREDITS ACHIEVED TOWARDS THE
National Certificate in Business Administration and Computing Level 2

Date: 26/09/2004 Brooke Diane (1234)

CERTIFICATE SUMMARY

Compulsory Requirements	Minimum Required	Achieved	Status
Domain Requirements:			
business administration services	10	24	Completed
business information processing	10	42	Completed
generic computing	9	27	Completed
interpersonal communications	5	0	
service sector - core skills	4	0	
Unit Standard Requirements:			
497 Protect health and safety in the wo	1	0	
Total Credits Required	60	93	Completed
Total Credits @ Level 2 required	40	68	Completed
Total Credits @ Level 1 Allowed	20	25	Completed

CERTIFICATE NOT YET ACHIEVED

Student Achievements and Options

Unit Standards within Optional Subfields

Subfield Business Administration
Domain business administration services

Unit No	Unit Standard Title	Level	Credits	Status
121	Operate clerical systems and apply busin	2	5	Completed
327	Document business transactions	2	3	Completed
328	Identify the requirements for a financia	2	3	Completed
329	Process financial information for cash t	2	3	Completed
331	Operate computer accounts receivable and	2	5	Completed
332	Operate computer general ledger accounti	2	5	Completed
Total Credits Achieved in this Domain			24	
Total Credits Available in Domain			24	

Unit Standards within Optional Subfields

Subfield Business Administration
Domain business information processing

Unit No	Unit Standard Title	Level	Credits	Status
101	Develop and use keyboarding skills to en	1	3	Completed
102	Consolidate keyboarding skills and devel	1	3	Completed
103	Use data entry skills to input computer	2	2	Completed
104	Introduce and apply audio transcription	2	2	Completed
107	Produce text processed communications	2	5	Completed
111	Operate a word processor	2	5	Completed
115	Write at 40 words per minute (wpm) and t	1	10	Completed
116	Write shorthand at 60 words per minute (	2	10	Completed
16677	Key in text at 15 words per minute (wpm)	1	1	Completed
16678	Key in text at 25 words per minute (wpm)	2	1	Completed

Total Credits Achieved in this Domain 42
Total Credits Available in Domain 42

Unit Standards within Optional Subfields

Subfield Business Administration
Domain text and information management - generic

Unit No	Unit Standard Title	Level	Credits	Status
12883	Enter and manage text for generic text a	1	3	
12884	Create documents and manage files for ge	2	3	
12885	Create and enhance documents combining t	2	6	

Total Credits Achieved in this Domain 0
Total Credits Available in Domain 12

Unit Standards within Optional Subfields

Subfield Communication Skills
Domain interpersonal communications

Unit No	Unit Standard Title	Level	Credits	Status
1277	Communicate information in a specified w	2	3	
1285	Make inquiries and complete practical tr	1	4	
1293	Participate in an informal one-to-one fa	1	2	
1294	Be interviewed in a formal situation	2	2	
1299	Be assertive in a range of specified sit	2	4	
1304	Communicate with people from other cultu	2	2	
3501	Apply listening techniques	1	4	
3503	Participate in a team or group to comple	1	2	
9677	Participate in groups and/or teams to ga	2	3	
9680	Communicate within an organisational con	2	2	
9705	Give and receive feedback	3	4	
9707	Demonstrate knowledge of workplace commu	1	5	
10790	Hold a conversation with others	1	2	
10791	Participate in informal meetings	2	3	

Total Credits Achieved in this Domain 0
Total Credits Available in Domain 42

Unit Standards within Optional Subfields

Subfield Core Generic
Domain work and study skills

Unit No	Unit Standard Title	Level	Credits	Status
1840	Practice accountability in the workplace	2	3	
1979	Describe the employment relationship, an	2	3	
1980	Identify, from an employee perspective,	3	3	
1982	Contribute to collective employment agre	3	3	
4251	Manage own career development	3	3	
4252	Produce a targeted resume	2	2	
4253	Obtain job search skills	2	3	
7117	Produce a plan to enhance own learning	2	2	
7118	Manage own learning programme	2	3	
7119	Describe memory processes and demonstrat	2	1	
7120	Demonstrate knowledge of written and vis	2	2	
7121	Demonstrate information search, access,	2	2	
7122	Organise, store, and retrieve informatio	2	3	
8557	Seek changes to an individual employment	2	3	
10781	Produce a plan for own future directions	2	3	
12360	Describe and explain emerging patterns o	2	2	
12381	Describe the role of unions in New Zeala	2	1	
12383	Explore career options relevant to an ar	2	2	
12384	Demonstrate knowledge of different think	3	3	
16688	Identify and manage the effects of shift	2	2	
20587	Apply knowledge of the Neuro-Linguistic	3	3	
20588	Apply knowledge of the Left Brain/Right	3	3	
20589	Apply knowledge of the Experiential Lear	3	3	

Total Credits Achieved in this Domain 0
Total Credits Available in Domain 58

Unit Standards within Optional Subfields

Subfield Core Generic
Domain self-management

Unit No	Unit Standard Title	Level	Credits	Status
496	Manage personal wellness	1	2	
524	Operate a personal budget	2	2	
548	Plan to manage personal use of alcohol a	1	2	
1827	Identify support services and resources	2	2	
4255	Demonstrate knowledge of personal insura	2	3	
4256	Demonstrate knowledge of personal financ	2	2	
4258	Describe ways of managing and coping wit	2	2	
7123	Demonstrate knowledge of problem solving	2	2	
7127	Exercise informed choice in deciding on	2	2	
8548	Demonstrate knowledge of accessing legal	1	2	
8549	Describe roles and expectations for part	1	2	
8553	Demonstrate a Disputes Tribunal presenta	2	2	
12348	Demonstrate knowledge of anger and optio	1	2	
12349	Demonstrate time management	2	3	
12352	Describe aspects of one's own lineage, h	2	3	
12353	Use personal banking services and person	2	3	
12354	Describe implications of independent liv	2	4	
12355	Demonstrate knowledge of stress and ways	2	2	
12357	Demonstrate knowledge of human sexuality	1	4	
12358	Demonstrate knowledge required for an in	1	3	
12359	Describe household conservation strategi	2	3	

Total Credits Achieved in this Domain 0
Total Credits Available in Domain 52

Domain Requirements and Options business administration services

Unit No	Unit Standard Title	Level	Credits	Status
121	Operate clerical systems and apply busin	2	5	Completed
327	Document business transactions	2	3	Completed
328	Identify the requirements for a financia	2	3	Completed
329	Process financial information for cash t	2	3	Completed
331	Operate computer accounts receivable and	2	5	Completed
332	Operate computer general ledger accounti	2	5	Completed

Total Credits for business administration services 24
Total Credits Required (if any) 10

Domain Requirements and Options business information processing

Unit No	Unit Standard Title	Level	Credits	Status
101	Develop and use keyboarding skills to en	1	3	Completed
102	Consolidate keyboarding skills and devel	1	3	Completed
103	Use data entry skills to input computer	2	2	Completed
104	Introduce and apply audio transcription	2	2	Completed
107	Produce text processed communications	2	5	Completed
111	Operate a word processor	2	5	Completed
115	Write at 40 words per minute (wpm) and t	1	10	Completed
116	Write shorthand at 60 words per minute (	2	10	Completed
16677	Key in text at 15 words per minute (wpm)	1	1	Completed
16678	Key in text at 25 words per minute (wpm)	2	1	Completed

Total Credits for business information processing 42
Total Credits Required (if any) 10

Domain Requirements and Options generic computing

Unit No	Unit Standard Title	Level	Credits	Status
112	Produce information using word processor	3	5	
2780	Operate and maintain a personal computer	2	9	
2781	Manage and protect data in a personal co	2	3	
2784	Create and use a simple computer spreads	2	3	Completed
2786	Create and use a simple computer flatfil	2	3	Completed
2788	Produce a simple desktop published docum	2	3	
2790	Use and maintain personal computer perip	2	3	
2791	Integrate spreadsheet and database data	2	3	
2792	Produce computer graphic documents using	1	2	Completed
2798	Demonstrate knowledge of the application	2	2	
5938	Access computer resources via a local ar	2	2	Completed
5939	Produce computer graphics using base fun	2	3	
5941	Exchange messages using electronic mail	2	2	
5942	Exchange information with an on-line com	2	3	
5943	Investigate and operate leisure-oriented	2	3	
5946	Use computer technology to present a com	2	3	
5949	Describe the management and use of compu	2	2	
5957	Produce simple schematic diagrams using	2	2	
5958	Produce a simple design illustration usi	2	3	
5962	Digitise text	2	2	
5966	Describe the computer control of mechani	2	3	
6743	Demonstrate an understanding of ergonomi	2	2	Completed
15167	Create individual web pages as a basis o	2	2	Completed
15168	Enhance pages on a website	2	4	
15169	Develop and publish an operational websi	2	3	
18734	Create a web homepage using a template	1	2	Completed
18735	Create a simple website to meet the spec	2	3	Completed
18736	Create simple webpages using a text edit	2	4	Completed
18743	Produce and use a spreadsheet	1	2	Completed
18758	Use a browser to navigate the world wide	1	2	Completed

Total Credits for generic computing 27
Total Credits Required (if any) 9

Domain Requirements and Options interpersonal communications

Unit No	Unit Standard Title	Level	Credits	Status
1277	Communicate information in a specified w	2	3	
1285	Make inquiries and complete practical tr	1	4	
1293	Participate in an informal one-to-one fa	1	2	
1294	Be interviewed in a formal situation	2	2	
1299	Be assertive in a range of specified sit	2	4	
1304	Communicate with people from other cultu	2	2	
3501	Apply listening techniques	1	4	
3503	Participate in a team or group to comple	1	2	
9677	Participate in groups and/or teams to ga	2	3	
9680	Communicate within an organisational con	2	2	
9705	Give and receive feedback	3	4	
9707	Demonstrate knowledge of workplace commu	1	5	
10790	Hold a conversation with others	1	2	
10791	Participate in informal meetings	2	3	

Total Credits for interpersonal communications 0
Total Credits Required (if any) 5

Domain Requirements and Options service sector - core skills

Unit No	Unit Standard Title	Level	Credits	Status
56	Attend to customer enquiries face-to-fac	1	2	
57	Provide customer service in given situat	2	2	
62	Maintain personal presentation in the wo	2	2	
64	Perform calculations for the workplace	1	2	
377	Work in a diverse workplace	2	2	

Total Credits for service sector - core skills 0
Total Credits Required (if any) 4

Domain Requirements and Options accounting - generic

Unit No	Unit Standard Title	Level	Credits	Status
7359	Explain the nature of accounting and bus	1	2	
7360	Process financial information to a trial	1	3	
7361	Prepare financial statements for a sole	1	4	
7362	Analyse, interpret, and present the fina	1	4	
7363	Process financial information using a ca	1	3	
7364	Manage personal finances	1	3	
7365	Examine skills and factors in operating	1	3	
7366	Review the control and accounting proced	2	4	
7367	Review the control and accounting proced	2	4	
7368	Review the control and accounting proced	2	4	
7369	Review the control and accounting proced	2	4	
7370	Review the control and accounting proced	2	4	
7371	Review the control and accounting proced	2	3	
7372	Explain and make balance day adjustments	2	2	
7373	Account for community organisations	2	3	

Total Credits for accounting - generic 0
Total Credits Required (if any) 0

Unit Standard Options and Achievements

Unit No	Unit Standard Title	Level	Credits	Status
497	Protect health and safety in the workpla	1	1	
Total Credits for these Unit Standards			0	

END OF PROGRESS REPORT

SUMMARY OF CREDITS ACHIEVED TOWARDS THE
National Certificate in Business Administration and Computing Level 3

Date : 26/09/2004 Brooke Diane (1234)

CERTIFICATE SUMMARY

Compulsory Requirements	Minimum Required	Achieved	Status
Subfield Requirements:			
Business Administration	25	78	Completed
Communication Skills	8	19	Completed
Domain Requirements:			
generic computing	10	88	Completed
service sector - core skills	2	0	
Total Credits Required	60	302	Completed
Total Credits @ Level 3 required	40	21	
Total Credits @ Level 2 Allowed	20	220	Completed
Total Credits @ Level 1 Allowed	0	61	Completed

CERTIFICATE NOT YET ACHIEVED

Student Achievements and Options

Unit Standards within Required Subfields

Subfield Business Administration
Domain text and information management - generic

Unit No	Unit Standard Title	Level	Credits	Status
12883	Enter and manage text for generic text a	1	3	Completed
12884	Create documents and manage files for ge	2	3	Completed
12885	Create and enhance documents combining t	2	6	Completed
Total Credits Achieved in this Domain			12	
Total Credits Available in Domain			12	

Unit Standards within Required Subfields

Subfield Business Administration
Domain business information processing

Unit No	Unit Standard Title	Level	Credits	Status
101	Develop and use keyboarding skills to en	1	3	Completed
102	Consolidate keyboarding skills and devel	1	3	Completed
103	Use data entry skills to input computer	2	2	Completed
104	Introduce and apply audio transcription	2	2	Completed
107	Produce text processed communications	2	5	Completed
111	Operate a word processor	2	5	Completed
115	Write at 40 words per minute (wpm) and t	1	10	Completed
116	Write shorthand at 60 words per minute (	2	10	Completed
16677	Key in text at 15 words per minute (wpm)	1	1	Completed
16678	Key in text at 25 words per minute (wpm)	2	1	Completed

Total Credits Achieved in this Domain 42
Total Credits Available in Domain 42

Unit Standards within Required Subfields

Subfield Business Administration
Domain business administration services

Unit No	Unit Standard Title	Level	Credits	Status
121	Operate clerical systems and apply busin	2	5	Completed
327	Document business transactions	2	3	Completed
328	Identify the requirements for a financia	2	3	Completed
329	Process financial information for cash t	2	3	Completed
331	Operate computer accounts receivable and	2	5	Completed
332	Operate computer general ledger accounti	2	5	Completed

Total Credits Achieved in this Domain 24
Total Credits Available in Domain 24

Unit Standards within Required Subfields

Subfield Communication Skills
Domain interpersonal communications

Unit No	Unit Standard Title	Level	Credits	Status
1277	Communicate information in a specified w	2	3	
1285	Make inquiries and complete practical tr	1	4	Completed
1293	Participate in an informal one-to-one fa	1	2	
1294	Be interviewed in a formal situation	2	2	
1299	Be assertive in a range of specified sit	2	4	
1304	Communicate with people from other cultu	2	2	
3501	Apply listening techniques	1	4	Completed
3503	Participate in a team or group to comple	1	2	
9677	Participate in groups and/or teams to ga	2	3	
9680	Communicate within an organisational con	2	2	
9705	Give and receive feedback	3	4	Completed
9707	Demonstrate knowledge of workplace commu	1	5	Completed
10790	Hold a conversation with others	1	2	Completed
10791	Participate in informal meetings	2	3	

Total Credits Achieved in this Domain 19
Total Credits Available in Domain 42

Unit Standards within Optional Subfields

Subfield Accounting
Domain accounting - generic

Unit No	Unit Standard Title	Level	Credits	Status
7359	Explain the nature of accounting and bus	1	2	
7360	Process financial information to a trial	1	3	
7361	Prepare financial statements for a sole	1	4	
7362	Analyse, interpret, and present the fina	1	4	
7363	Process financial information using a ca	1	3	
7364	Manage personal finances	1	3	
7365	Examine skills and factors in operating	1	3	
7366	Review the control and accounting proced	2	4	
7367	Review the control and accounting proced	2	4	Completed
7368	Review the control and accounting proced	2	4	Completed
7369	Review the control and accounting proced	2	4	Completed
7370	Review the control and accounting proced	2	4	
7371	Review the control and accounting proced	2	3	Completed
7372	Explain and make balance day adjustments	2	2	
7373	Account for community organisations	2	3	

Total Credits Achieved in this Domain 15
Total Credits Available in Domain 50

Unit Standards within Optional Subfields

Subfield Financial Management
Domain credit administration

Unit No	Unit Standard Title	Level	Credits	Status
16757	Determine applicant's suitability for cr	3	8	
16758	Administer debt collection	3	8	
16759	Demonstrate telephone techniques for deb	3	6	
Total Credits Achieved in this Domain			0	
Total Credits Available in Domain			22	

Unit Standards within Optional Subfields

Subfield Management
Domain quality management

Unit No	Unit Standard Title	Level	Credits	Status
8081	Collect data for the management of quali	3	8	
8085	Explain fundamental concepts and princip	3	4	
8086	Demonstrate knowledge required for quali	3	4	
8087	Use core quality management tools	3	6	
Total Credits Achieved in this Domain			0	
Total Credits Available in Domain			22	

Domain Requirements and Options generic computing

Unit No	Unit Standard Title	Level	Credits	Status
112	Produce information using word processor	3	5	Completed
2780	Operate and maintain a personal computer	2	9	Completed
2781	Manage and protect data in a personal co	2	3	Completed
2784	Create and use a simple computer spreads	2	3	Completed
2786	Create and use a simple computer flatfil	2	3	Completed
2788	Produce a simple desktop published docum	2	3	Completed
2790	Use and maintain personal computer perip	2	3	Completed
2791	Integrate spreadsheet and database data	2	3	Completed
2792	Produce computer graphic documents using	1	2	Completed
2798	Demonstrate knowledge of the application	2	2	Completed
5938	Access computer resources via a local ar	2	2	Completed
5939	Produce computer graphics using base fun	2	3	Completed
5941	Exchange messages using electronic mail	2	2	Completed
5942	Exchange information with an on-line com	2	3	Completed
5943	Investigate and operate leisure-oriented	2	3	Completed
5946	Use computer technology to present a com	2	3	Completed
5949	Describe the management and use of compu	2	2	Completed
5957	Produce simple schematic diagrams using	2	2	Completed
5958	Produce a simple design illustration usi	2	3	Completed
5962	Digitise text	2	2	Completed
5966	Describe the computer control of mechani	2	3	Completed
6743	Demonstrate an understanding of ergonomi	2	2	Completed
15167	Create individual web pages as a basis o	2	2	Completed

15168	Enhance pages on a website	2	4	Completed
15169	Develop and publish an operational websi	2	3	Completed
18734	Create a web homepage using a template	1	2	Completed
18735	Create a simple website to meet the spec	2	3	Completed
18736	Create simple webpages using a text edit	2	4	Completed
18743	Produce and use a spreadsheet	1	2	Completed
18758	Use a browser to navigate the world wide	1	2	Completed

Total Credits for generic computing 88
Total Credits Required (if any) 10

Domain Requirements and Options service sector - core skills

Unit No	Unit Standard Title	Level	Credits	Status
56	Attend to customer enquiries face-to-fac	1	2	
57	Provide customer service in given situat	2	2	
62	Maintain personal presentation in the wo	2	2	
64	Perform calculations for the workplace	1	2	
377	Work in a diverse workplace	2	2	

Total Credits for service sector - core skills 0
Total Credits Required (if any) 2

Domain Requirements and Options

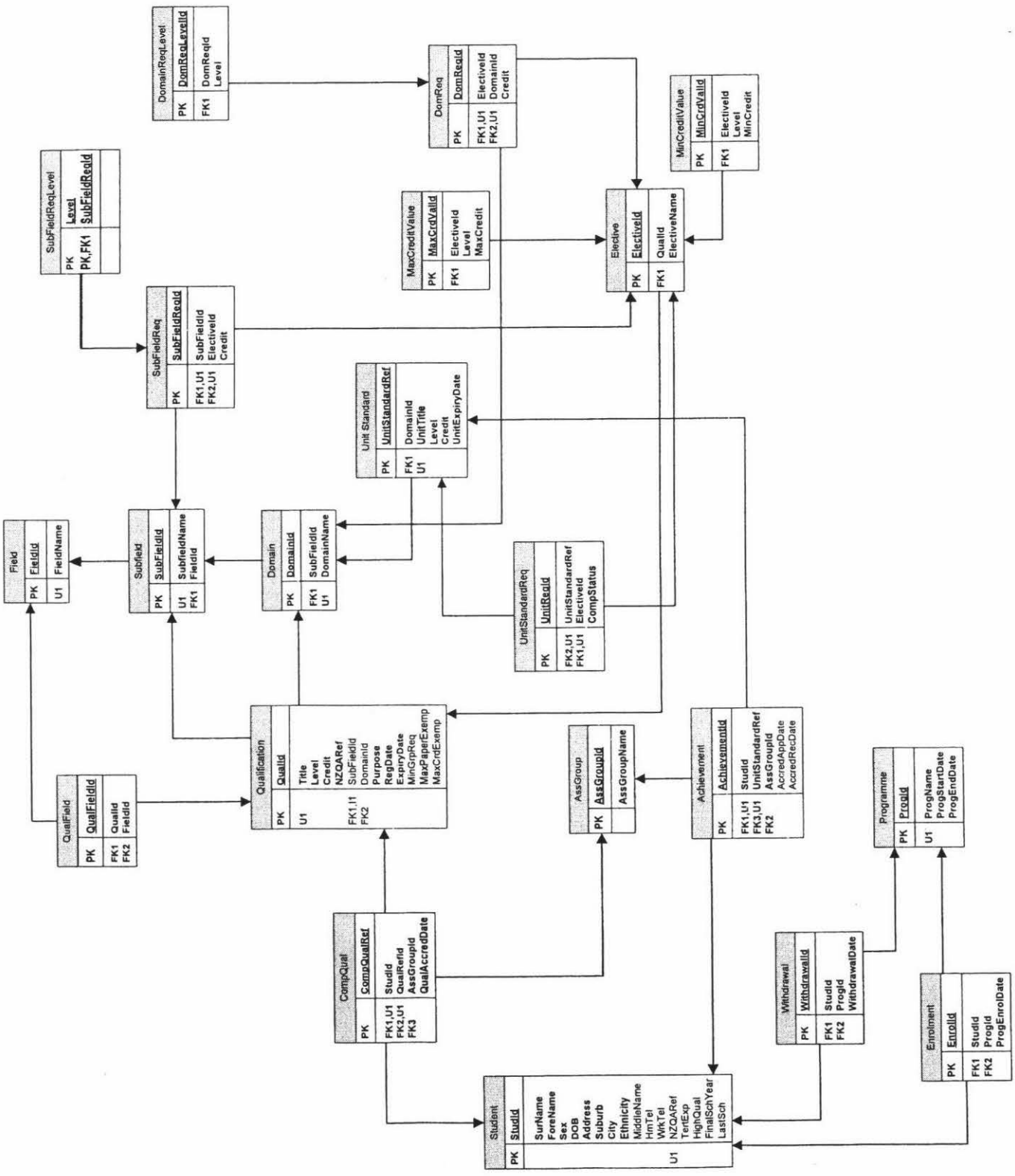
Required at Level 3 or more for Domain work and study skills

Unit No	Unit Standard Title	Level	Credits	Status
1980	Identify, from an employee perspective,	3	3	
1982	Contribute to collective employment agre	3	3	Completed
4251	Manage own career development	3	3	
12384	Demonstrate knowledge of different think	3	3	
20587	Apply knowledge of the Neuro-Linguistic	3	3	Completed
20588	Apply knowledge of the Left Brain/Right	3	3	Completed
20589	Apply knowledge of the Experiential Lear	3	3	Completed

Total Credits for work and study skills 12
Total Credits Required at Level 3 or more (if any) 0

END OF PROGRESS REPORT

Appendix 7 – Logical Model for Student Mentoring System



```
                                    CONCPLOG.DDL
/*    This SQL DDL script was generated by Microsoft Visual Studio (Release
Date: LOCAL BUILD). */

/*    Driver Used : Microsoft Visual Studio - Microsoft SQL Server Driver.
                  */
/*    Document    : C:\DOCUMENTS AND SETTINGS\ADRIAN\MY
DOCUMENTS\WORK\MASTERS\CASESTUDY\VBPROTO\CONCPLOG.VSD. */
/*    Time Created: 1 August 2004 7:46 p.m..
                  */
/*    Operation   : From Visio Generate Wizard.
                  */
/*    Connected data source : No connection.
                  */
/*    Connected server      : No connection.
                  */
/*    Connected database    : Not applicable.
                  */



/* Create Mentor database.
                  */
use master

go

create database "Mentor"

go

use "Mentor"

go

/* Create new table "Withdrawal".
                  */
/* "Withdrawal" : Table of Student withdraws from Programme Date
                  */
/*      "WithdrawalId" : WithdrawalId identifies Student withdraws from
Programme Date            */
/*      "StudId" : Student is a learner who is enrolled in a Programme of
study at McGirr Training. */
/*      "ProgId" : Programme is a programme of study undertaken by a
student at McGirr Training.  */
/*      "WithdrawalDate" : Date is the day, month and year specified in the
fact that something has occurred. */
create table "Withdrawal" (
        "WithdrawalId" int not null,
        "StudId" int not null,
        "ProgId" int not null,
        "WithdrawalDate" smalldatetime not null)

go

alter table "Withdrawal"
        add constraint "Withdrawal_PK" primary key ("WithdrawalId")


go

/* Create new table "Enrolment".
                  */
/* "Enrolment" : Table of Student enrols Programme Date
                  */
/*      "EnrolId" : EnrolId identifies Student enrols Programme Date
                     */
/*      "StudId" : Student is a learner who is enrolled in a Programme of
study at McGirr Training. */
/*      "ProgId" : Programme is a programme of study undertaken by a
student at McGirr Training.  */
/*      "ProgEnrolDate" : Date is the day, month and year specified in the
```

```
fact that something has occurred. */
create table "Enrolment" (
        "EnrolId" int not null,
        "StudId" int not null,
        "ProgId" int not null,
        "ProgEnrolDate" smalldatetime not null)

go

alter table "Enrolment"
        add constraint "Enrolment_PK" primary key ("EnrolId")


go

/* Create new table "DomReq".
                    */
/* "DomReq" : Domain Requirement is a requirement specified in an Elective
of a Qualification for a certain number of Unit Standard credits from a
Domain. */
/*      "DomReqId" : Domain Requirement is a requirement specified in an
Elective of a Qualification for a certain number of Unit Standard credits
from a Domain. */
/*      "ElecId" : Elective is a specified set of requirements associated
with a particular Qualification. Each Elective specifies the Subfield or
Domain from which Unit Standards may (or must) be achieved to complete that
Elective. The Elective may also specify a minimum Level and/or Credit Value
that Unit Standards must meet to achieve the Elective.The Elective may also
specify a particular Unit Standard(s) that may (or must) be achieved. The
Elective can be compulsory or optional. An Elective may also be called a
'Strand' in some Qualification specifications. */
/*      "DomId" : Domain is an an area of learning within a SubField on the
National Qualifications Framework, as defined in the National
Qualifications Framework classification system. */
/*      "Credit" : Credit Value is a numerical value assigned to a Unit
Standard on the National Qualifications Framework that represents the
estimated time in hours needed for a typical learner to demonstrate that
all specified outcomes have been met. */
create table "DomReq" (
        "DomReqId" int not null,
        "ElecId" int not null,
        "DomId" int not null,
        "Credit" int not null constraint "DomReqCredit_Chk" check ("Credit"
between 1 and 20) )

go

alter table "DomReq"
        add constraint "DomReq_PK" primary key ("DomReqId")


go

/* Create new table "CompUnitStand".
                    */
/* "CompUnitStand" : Unit Standard is a nationally registered, coherent set
of learning outcomes and associated performance criteria, together with
technical and management information that supports delivery and assessment.
All unit standards are registered on the National Qualifications Framework,
assigned a level and a credit value, and may contribute to the award of a
National Certificate or Diploma.  */
/*      "CompUnitRef" : Unit Standard is a nationally registered, coherent
set of learning outcomes and associated performance criteria, together with
technical and management information that supports delivery and assessment.
All unit standards are registered on the National Qualifications Framework,
assigned a level and a credit value, and may contribute to the award of a
National Certificate or Diploma.  */
/*      "StudId" : Student is a learner who is enrolled in a Programme of
study at McGirr Training. */
/*      "UnitRef" : Date is the day, month and year specified in the fact
that something has occurred. */
```

```
/*      "AssGroupId" : Assessment Group is a group of Unit Standards
achievements for a particular Student. Assessment Groups are used to match
against a set of Qualification requirements to determine whether they have
been met. */
/*      "UntAccredDate" : Date is the day, month and year specified in the
fact that something has occurred. */
create table "CompUnitStand" (
        "CompUnitRef" int not null,
        "StudId" int not null,
        "UnitRef" int not null,
        "AssGroupId" int not null,
        "UntAccredDate" smalldatetime not null)

go

alter table "CompUnitStand"
        add constraint "CompUnitStand_PK" primary key ("CompUnitRef")


go

/* Create new table "CompQual".
                */
/* "CompQual" : Completed Qualification is a Qualification achieved by a
student prior to Enrolment onto a Mcgirr Training Programme. */
/*      "CompQualRef" : Completed Qualification is a Qualification achieved
by a student prior to Enrolment onto a Mcgirr Training Programme. */
/*      "StudId" : Student is a learner who is enrolled in a Programme of
study at McGirr Training. */
/*      "QualRef" : Completed Qualification is a Qualification achieved by
a student prior to Enrolment onto a Mcgirr Training Programme. */
/*      "AssGroupId" : Assessment Group is a group of Unit Standards
achievements for a particular Student. Assessment Groups are used to match
against a set of Qualification requirements to determine whether they have
been met. */
/*      "QualAccredDate" : Date is the day, month and year specified in the
fact that something has occurred. */
create table "CompQual" (
        "CompQualRef" int not null,
        "StudId" int not null,
        "QualRef" int not null,
        "AssGroupId" int not null,
        "QualAccredDate" smalldatetime not null)

go

alter table "CompQual"
        add constraint "CompQual_PK" primary key ("CompQualRef")


go

/* Create new table "QualField".
                */
/* "QualField" : Records those qualifications that exist at the field or
multi-field level on the NZQA framework */
/*      "QualfieldId" : QualfieldId partly identifies QualificationField
                */
/*      "QualRefId" : qualRef is of QualificationField
                */
/*      "FieldId" : fldId is of QualificationField
                */
create table "QualField" (
        "QualfieldId" int not null,
        "QualRefId" int not null,
        "FieldId" int not null)

go

alter table "QualField"
        add constraint "Qual Field_PK" primary key ("QualfieldId")
```

```
go

/* Create new table "SubFieldReq".
                    */
/* "SubFieldReq" : A qualification requirement for a particular elective at
the subfield level i.e. used to record a requirement in which any unit
standard(s) within a particular subfield can be used to meet that
requirement. */
/*      "SubFieldReqId" : SubField Requirement is a requirement specified
in an Elective of a Qualification for a certain number of Unit Standard
credits from a SubField */
/*      "SubFieldId" : Subfield is an area of learning within a particular
Field on the National Qualifications Framework. */
/*      "ElectiveId" : Elective is a specified set of requirements
associated with a particular Qualification. Each Elective specifies the
Subfield or Domain from which Unit Standards may (or must) be achieved to
complete that Elective. The Elective may also specify a minimum Level
and/or Credit Value that Unit Standards must meet to achieve the
Elective.The Elective may also specify a particular Unit Standard(s) that
may (or must) be achieved. The Elective can be compulsory or optional. An
Elective may also be called a 'Strand' in some Qualification
specifications. */
/*      "Credit" : Credit Value is a numerical value assigned to a Unit
Standard on the National Qualifications Framework that represents the
estimated time in hours needed for a typical learner to demonstrate that
all specified outcomes have been met. */
create table "SubFieldReq" (
        "SubFieldReqId" int not null,
        "SubFieldId" int not null,
        "ElectiveId" int not null,
        "Credit" int not null constraint "SubFieldReqCredit_Chk" check
("Credit" between 1 and 20) )

go

alter table "SubFieldReq"
        add constraint "SubFieldReq_PK" primary key ("SubFieldReqId")


go

/* Create new table "Subfield".
                    */
/* "Subfield" : Records each subfield within the NZQA framework
                    */
/*      "SubFieldId" : Subfield is an area of learning within a particular
Field on the National Qualifications Framework. */
/*      "SubfieldName" : Subfield Name is the name given to a particular
Subfield as defined in the National Qualifications Framework classification
system. */
/*      "FieldId" : Field is a broad area of learning on the National
Qualifications Framework, as defined in the National Qualifications
Framework classification system. */
create table "Subfield" (
        "SubFieldId" int not null,
        "SubfieldName" varchar(30) not null,
        "FieldId" int not null)

go

alter table "Subfield"
        add constraint "SubField_PK" primary key ("SubFieldId")


go

/* Create new table "Qualification".
                    */
/* "Qualification" : Records each qualification within the NZQA framework.
```

```
                    */
/*      "QualRefId" : Completed Qualification is a Qualification achieved
by a student prior to Enrolment onto a Mcgirr Training Programme. */
/*      "Title" : Title is a name assigned to a Qualification on the
National Qualification Framework. */
/*      "Level" : Level is a descriptor that indicates the complexity of
learning associated with each Qualification. There are ten levels involved
in a Qualification - 1 is the least complex and 10 the most. They do not
equate to 'years spent learning' but reflect the content of the
Qualification. */
/*      "Credit" : Credit Value is a numerical value assigned to a Unit
Standard on the National Qualifications Framework that represents the
estimated time in hours needed for a typical learner to demonstrate that
all specified outcomes have been met. */
/*      "NZQARef" : Qualification has NQF Reference
                    */
/*      "SubFieldId" : Subfield is an area of learning within a particular
Field on the National Qualifications Framework. */
/*      "DomainId" : Domain is an an area of learning within a SubField on
the National Qualifications Framework, as defined in the National
Qualifications Framework classification system. */
/*      "Purpose" : Purpose is a statement issued by the NZQA as to the
intention of a Qualification. Details include an exit profile of a typical
student who has completed the Qualification. */
/*      "RegDate" : Date is the day, month and year specified in the fact
that something has occurred. */
/*      "ExpiryDate" : Date is the day, month and year specified in the
fact that something has occurred. */
/*      "MinGrpReq" : Certain qualifications at level 4 and above specify a
minimum number of groups from which a student can select to meet specific
requirements for that qualification. */
/*      "MaxPaperExemp" : Maximum number of papers
                    */
/*      "MaxCrdExemp" : Maximum Credit Exemption is  the maximum number of
credits that may be granted towards a qualification on the basis of credits
already earned in another qualification. Credits have to be recognised
before they are transferred. */
create table "Qualification" (
        "QualRefId" int not null,
        "Title" varchar(40) not null,
        "Level" int not null constraint "QualificationLevel_Chk" check
("QualLevel" between 1 and 5) ,
        "Credit" int not null constraint "QualificationCredit_Chk" check
("Credit" between 1 and 20) ,
        "NZQARef" char(10) not null constraint "QualificationNZQARef_Chk"
check ( not null ) ,
        "SubFieldId" int null,
        "DomainId" int null,
        "Purpose" text not null,
        "RegDate" smalldatetime not null,
        "ExpiryDate" smalldatetime not null,
        "MinGrpReq" int null,
        "MaxPaperExemp" int null constraint
"QualificationMaxPaperExemp_Chk" check ("MaxPaperExemp" between 1 and 10) ,
        "MaxCrdExemp" int null constraint "QualificationMaxCrdExemp_Chk"
check ("MaxCrdExemp" between 1 and 20) )

go

alter table "Qualification"
        add constraint "Qualification_PK" primary key ("QualRefId")


go

/* Create new table "Domain".
                    */
/* "Domain" : Records each Domain within the NZQA framework.
                    */
/*      "DomainId" : Domain is an an area of learning within a SubField on
the National Qualifications Framework, as defined in the National
```

```
Qualifications Framework classification system. */
/*      "SubFieldId" : Subfield is an area of learning within a particular
Field on the National Qualifications Framework. */
/*      "DomainName" : Domain Name is the name given to a particular Domain
as defined in the National Qualifications Framework classification system.
*/
create table "Domain" (
        "DomainId" int not null,
        "SubFieldId" int not null,
        "DomainName" varchar(30) not null)

go

alter table "Domain"
        add constraint "Domain_PK" primary key ("DomainId")


go

/* Create new table "MaxCreditValue".
                */
/* "MaxCreditValue" : Each elective has a maximum credit value at a
particular level (1-7). This detail is often shown on the first page of
each qualification word document on the NZQA website. It is often shown
together with the minimum credit value for that elective at a particular
level  i.e. a credit value range. Maximum credit value is the upper limit
of that range. */
/*      "MaxCrdValId" : MaxCrdValId partly identifies MaxCreditValue
                        */
/*      "ElectiveId" : elecId is of MaxCreditValue
                        */
/*      "Level" : Level is a descriptor that indicates the complexity of
learning associated with each Qualification. There are ten levels involved
in a Qualification - 1 is the least complex and 10 the most. They do not
equate to 'years spent learning' but reflect the content of the
Qualification. */
/*      "MaxCredit" : Maximum Credit Value is the minimum number of Unit
Standard credits allowed for this Elective at this Level. This detail is
often shown on the first page of each qualification word document on the
NZQA website. It is often shown together with the minimum credit value for
that elective at a particular level i.e. a credit value range. Maximum
Credit Value is the upper limit of that range. */
create table "MaxCreditValue" (
        "MaxCrdValId" int not null,
        "ElectiveId" int not null,
        "Level" int not null constraint "MaxCreditValueLevel_Chk" check
("Level" between 1 and 8) ,
        "MaxCredit" int not null constraint "MaxCreditValueMaxCredit_Chk"
check ("MaxCredit" between 1 and 100) )

go

alter table "MaxCreditValue"
        add constraint "MaxCreditValue_PK" primary key ("MaxCrdValId")


go

/* Create new table "MinCreditValue".
                */
/* "MinCreditValue" : Each elective has a minimum credit value at a
particular level (1-7). This detail is often shown on the first page of
each qualification word document on the NZQA website. It is often shown
together with the maximum credit value for that elective at a particular
level  i.e. a credit value range. Mimimum credit value is the lower limit
of that range. */
/*      "MinCrdValId" : MinCrdValId partly identifies MinCreditValue
                        */
/*      "ElectiveId" : elecId is of MinCreditValue
                        */
/*      "MinCredit" : Minimum Credit Value is the minimum number of Unit
```

```
Standard credits allowed for this Elective at this Level. This detail is
often shown on the first page of each qualification word document on the
NZQA website. It is often shown together with the maximum credit value for
that elective at a particular level i.e. a credit value range. Mimimum
credit value is the lower limit of that range. */
/*      "Level" : Level is a descriptor that indicates the complexity of
learning associated with each Qualification. There are ten levels involved
in a Qualification - 1 is the least complex and 10 the most. They do not
equate to 'years spent learning' but reflect the content of the
Qualification. */
create table "MinCreditValue" (
        "MinCrdValId" int not null,
        "ElectiveId" int not null,
        "MinCredit" int not null constraint "MinCreditValueMinCredit_Chk"
check ("MinCredit" between 0 and 100) ,
        "Level" int not null constraint "MinCreditValueLevel_Chk" check
("Level" between 1 and 8) )

go

alter table "MinCreditValue"
        add constraint "MinCreditValue_PK" primary key ("MinCrdValId")


go

/* Create new table "Unit Standard".
                */
/* "Unit Standard" : Records each unit standard in the NZQA framework.
Obviously, this could include a large number of records and may be
restricted to those unit standards currently offered on Mcgirr training
programmes. */
/*      "UnitStandardRef" : Date is the day, month and year specified in
the fact that something has occurred. */
/*      "DomainId" : Domain is an an area of learning within a SubField on
the National Qualifications Framework, as defined in the National
Qualifications Framework classification system. */
/*      "UnitTitle" : Unit Title is the title given to a particular Unit
Standard as defined in the National Qualifications Framework classification
system. */
/*      "Level" : Level is a descriptor that indicates the complexity of
learning associated with each Qualification. There are ten levels involved
in a Qualification - 1 is the least complex and 10 the most. They do not
equate to 'years spent learning' but reflect the content of the
Qualification. */
/*      "Credit" : Credit Value is a numerical value assigned to a Unit
Standard on the National Qualifications Framework that represents the
estimated time in hours needed for a typical learner to demonstrate that
all specified outcomes have been met. */
/*      "UnitExpiryDate" : Date is the day, month and year specified in the
fact that something has occurred. */
create table "Unit Standard" (
        "UnitStandardRef" int not null,
        "DomainId" int not null,
        "UnitTitle" char(60) not null,
        "Level" int not null constraint "Unit StandardLevel_Chk" check
("Level" between 1 and 8) ,
        "Credit" int not null constraint "Unit StandardCredit_Chk" check
("Credit" between 1 and 20) ,
        "UnitExpiryDate" smalldatetime not null)

go

alter table "Unit Standard"
        add constraint "UnitStandard_PK" primary key ("UnitStandardRef")


go

/* Create new table "Student".
                */
```

```
/* "Student" : Record student details. Most but not all student details
have been included within this table since this table already exists in the
current database. */
/*      "StudId" : Student is a learner who is enrolled in a Programme of
study at McGirr Training. */
/*      "SurName" : Surname is the last name of the Student.
                        */
/*      "ForeName" : Forename is the first name of a Student.
                        */
/*      "Sex" : Sex is the gender of the Student.
                        */
/*      "DOB" : Date is the day, month and year specified in the fact that
something has occurred. */
/*      "Address" : Address is the street address of a Student.
                        */
/*      "Suburb" : Suburb is the suburb in which a Student lives.
                        */
/*      "City" : City is the City in which a Student lives.
                        */
/*      "Ethnicity" : Ethnicity is the racial affiliation of a Student.
                        */
/*      "MiddleName" : Middle Name is the middle name of a Student.
                        */
/*      "HmTel" : HomeTelephone is a Student's home telepone number
                        */
/*      "WrkTel" : Student has Work Telephone
                        */
/*      "NZQARef" : NZQA Reference is a unique reference number assigned to
a Student who has registered with the NZQA. */
/*      "TertExp" : Tertiary Experience is an idication of whether or not a
Student has studied within a post compulsory educational evironment e.g.
polytechnic, university or another P.T.E */
/*      "HighQual" : Highest Qualification is the highest educational
achivement of a Student.    */
/*      "FinalSchYear" : Student has Final School Year
                        */
/*      "LastSch" : Last School is the name of the last school attended by
a Student.              */
create table "Student" (
        "StudId" int not null,
        "SurName" varchar(30) not null,
        "ForeName" varchar(30) not null,
        "Sex" varchar(6) not null constraint "StudentSex_Chk" check ("Sex"
in ('Male','Female')) ,
        "DOB" smalldatetime not null,
        "Address" varchar(30) not null,
        "Suburb" varchar(30) not null,
        "City" varchar(30) not null,
        "Ethnicity" char(3) not null constraint "StudentEthnicity_Chk"
check ("Ethnicity" in
('NZM','EPK','OE','CIM','FIJ','NIU','SAM','TON','TOK','OPI','CHI','SEA','OA
','IND','OTH')) ,
        "MiddleName" varchar(30) null,
        "HmTel" varchar(15) null,
        "WrkTel" varchar(15) null,
        "NZQARef" varchar(15) null,
        "TertExp" char(10) null constraint "StudentTertExp_Chk" check
("TertiaryExp" in ('Yes','No')) ,
        "HighQual" varchar(5) null constraint "StudentHighQual_Chk" check
("HighQualAchieved" in ('UNQ','SCA','SFA','UE','DEG','TRC','Other')) ,
        "FinalSchYear" smalldatetime null,
        "LastSch" varchar(30) null)

go

alter table "Student"
        add constraint "Student_PK" primary key ("StudId")

go
```

```
/* Create new table "Field".
                */
/* "Field" : Records each field within the NZQA framework
                */
/*      "FieldId" : Field is a broad area of learning on the National
Qualifications Framework, as defined in the National Qualifications
Framework classification system. */
/*      "FieldName" : Field Name is the name given to a particular Field as
defined in the National Qualifications Framework classification system. */

create table "Field" (
        "FieldId" int not null,
        "FieldName" varchar(30) not null)

go

alter table "Field"
        add constraint "Field_PK" primary key ("FieldId")


go

/* Create new table "Elective".
                */
/* "Elective" : Each qualification consists of electives that in turn
consist of requirements. An elective can be compulsory or optional (perhaps
a field could be added to this table to indicate this). An elective could
also be a 'Strand' since qualification strands also consist of a set of
requirements. */
/*      "ElectiveId" : Elective is a specified set of requirements
associated with a particular Qualification. Each Elective specifies the
Subfield or Domain from which Unit Standards may (or must) be achieved to
complete that Elective. The Elective may also specify a minimum Level
and/or Credit Value that Unit Standards must meet to achieve the
Elective.The Elective may also specify a particular Unit Standard(s) that
may (or must) be achieved. The Elective can be compulsory or optional. An
Elective may also be called a 'Strand' in some Qualification
specifications. */
/*      "QualId" : Completed Qualification is a Qualification achieved by a
student prior to Enrolment onto a Mcgirr Training Programme. */
/*      "ElectiveName" : Elective Name is a name given to an Elective of a
particular Qualification as specified on the National Qualification
Framework's classification scheme. */
create table "Elective" (
        "ElectiveId" int not null,
        "QualId" int not null,
        "ElectiveName" varchar(30) not null)

go

alter table "Elective"
        add constraint "Elective_PK" primary key ("ElectiveId")


go

/* Create new table "Achievement".
                */
/* "Achievement" : Records each individual student achievement i.e. the
particular unit standard that has been completed. */
/*      "AchievementId" : Achievement is an individual student achievement
i.e. the particular unit standard that has been successfully completed. */
/*      "StudId" : Student is a learner who is enrolled in a Programme of
study at McGirr Training. */
/*      "AssGroupId" : Assessment Group is a group of Unit Standards
achievements for a particular Student. Assessment Groups are used to match
against a set of Qualification requirements to determine whether they have
been met. */
/*      "UnitStandardRef" : Date is the day, month and year specified in
the fact that something has occurred. */
/*      "AccredAppDate" : Date is the day, month and year specified in the
```

```
fact that something has occurred. */
/*      "AccredRecDate" : Date is the day, month and year specified in the
fact that something has occurred. */
create table "Achievement" (
        "AchievementId" int not null,
        "StudId" int not null,
        "AssGroupId" int not null,
        "UnitStandardRef" int not null,
        "AccredAppDate" smalldatetime null,
        "AccredRecDate" smalldatetime null)

go

alter table "Achievement"
        add constraint "Achievement_PK" primary key ("AchievementId")


go

/* Create new table "Programme".
                */
/* "Programme" : Programme relates to a programme of study undertaken by a
student             */
/*      "ProgId" : Programme is a programme of study undertaken by a
student at McGirr Training.  */
/*      "ProgName" : Programme Name is the name given to an area of study
at McgirrTraining.      */
/*      "ProgStartDate" : Date is the day, month and year specified in the
fact that something has occurred. */
/*      "ProgEndDate" : Date is the day, month and year specified in the
fact that something has occurred. */
create table "Programme" (
        "ProgId" int not null,
        "ProgName" varchar(30) not null,
        "ProgStartDate" smalldatetime not null,
        "ProgEndDate" smalldatetime not null)

go

alter table "Programme"
        add constraint "Programme_PK" primary key ("ProgId")


go

/* Create new table "UnitStandardReq".
                */
/* "UnitStandardReq" : A qualification requirement for a particular
elective at the unit standard level i.e. used to record a requirement in
which a specific unit standard can be used to meet that requirement. */
/*      "UnitReqId" : Unit Standard Requirement is a requirement (possibly
optional) specified in an Elective of a Qualification for a particular Unit
Standard. */
/*      "UnitStandardRef" : Date is the day, month and year specified in
the fact that something has occurred. */
/*      "ElectiveId" : Elective is a specified set of requirements
associated with a particular Qualification. Each Elective specifies the
Subfield or Domain from which Unit Standards may (or must) be achieved to
complete that Elective. The Elective may also specify a minimum Level
and/or Credit Value that Unit Standards must meet to achieve the
Elective.The Elective may also specify a particular Unit Standard(s) that
may (or must) be achieved. The Elective can be compulsory or optional. An
Elective may also be called a 'Strand' in some Qualification
specifications. */
/*      "CompStatus" : Compulsory Status is an indication of whether or not
a Unit Standard Requirement must be completed by a Student. */
create table "UnitStandardReq" (
        "UnitReqId" int not null,
        "UnitStandardRef" int not null,
        "ElectiveId" int not null,
        "CompStatus" char(3) not null constraint
```

```
"UnitStandardReqCompStatus_Chk" check ("CompStatus" in ('Yes','No')) )

go

alter table "UnitStandardReq"
        add constraint "UnitStandardReq_PK" primary key ("UnitReqId")


go

/* Create new table "DomainReqLevel".
                    */
/* "DomainReqLevel" : Each domain requirement can relate to unit standards
across a number of levels (1-7) e.g. 10 credits from the generic computing
domain at level 3 or above. This table records each level that a particular
domain requirement can exist. */
/*      "DomReqLevelId" : DomReqLevelId partly identifies DomainReqLevel
                        */
/*      "DomReqId" : domReqId is of DomainReqLevel
                        */
/*      "Level" : Level is a descriptor that indicates the complexity of
learning associated with each Qualification. There are ten levels involved
in a Qualification - 1 is the least complex and 10 the most. They do not
equate to 'years spent learning' but reflect the content of the
Qualification. */
create table "DomainReqLevel" (
        "DomReqLevelId" int not null,
        "DomReqId" int not null,
        "Level" int not null constraint "DomainReqLevelLevel_Chk" check
("Level" between 1 and 8) )

go

alter table "DomainReqLevel"
        add constraint "DomainReqLevel_PK" primary key ("DomReqLevelId")


go

/* Create new table "SubFieldReqLevel".
                    */
/* "SubFieldReqLevel" : Each subfield requirement can relate to unit
standards across a number of levels (1-7) e.g. 20 credits from the
computing subfield at level 3 or above. This table records each level that
a particular subfield requirement can exist. */
/*      "Level" : Level is a descriptor that indicates the complexity of
learning associated with each Qualification. There are ten levels involved
in a Qualification - 1 is the least complex and 10 the most. They do not
equate to 'years spent learning' but reflect the content of the
Qualification. */
/*      "SubFieldReqId" : SubField Requirement is a requirement specified
in an Elective of a Qualification for a certain number of Unit Standard
credits from a SubField */
create table "SubFieldReqLevel" (
        "Level" int not null constraint "SubFieldReqLevelLevel_Chk" check
("Level" between 1 and 8) ,
        "SubFieldReqId" int not null)

go

alter table "SubFieldReqLevel"
        add constraint "SubFieldReqLevel_PK" primary key ("SubFieldReqId",
"Level")


go

/* Create new table "AssGrpUnitReq".
                    */
/* "AssGrpUnitReq" : Used to record which particular assessment group meets
which particular Unitstandard Requirement for the student's current
```

```
qualification goal. */
/*      "AssGroupId" : assGrpId is of AssGrpUnitReq
                       */
/*      "UnitReqId" : untReqId is of AssGrpUnitReq
                       */
/*      "AssGrpUnitReqId" : AssGrpUnitReqId partly identifies AssGrpUnitReq
                       */
create table "AssGrpUnitReq" (
        "AssGroupId" int not null,
        "UnitReqId" int not null,
        "AssGrpUnitReqId" int not null)

go

alter table "AssGrpUnitReq"
        add constraint "AssGrpUnitReq_PK" primary key ("AssGrpUnitReqId")



go

/* Create new table "AssGrpDomainReq".
                  */
/* "AssGrpDomainReq" : Used to record which particular assessment group
meets which particular Domain Requirement for the student's current
qualification goal. */
/*      "AssGrpDomReqId" : AssGrpDomReqId partly identifies AssGrpDomainReq
                       */
/*      "AssGroupId" : assGrpId is of AssGrpDomainReq
                       */
/*      "DomReqId" : domReqId is of AssGrpDomainReq
                       */
create table "AssGrpDomainReq" (
        "AssGrpDomReqId" int not null,
        "AssGroupId" int not null,
        "DomReqId" int not null)

go

alter table "AssGrpDomainReq"
        add constraint "AssGrpDomainReq_PK" primary key ("AssGrpDomReqId")



go

/* Create new table "AssGrpSubFieldReq".
                  */
/* "AssGrpSubFieldReq" : Used to record which particular assessment group
meets which particular Subfield Requirement for the student's current
qualification goal. */
/*      "AssGrpSubFldReqId" : AssGrpSubFldReqId partly identifies
AssGrpSubFieldReq                  */
/*      "AssGroupId" : assGrpId is of AssGrpSubFieldReq
                       */
/*      "SubFieldReqId" : sbFldReqId is of AssGrpSubFieldReq
                       */
create table "AssGrpSubFieldReq" (
        "AssGrpSubFldReqId" int not null,
        "AssGroupId" int not null,
        "SubFieldReqId" int not null)

go

alter table "AssGrpSubFieldReq"
        add constraint "AssGrpSubFieldReq_PK" primary key
("AssGrpSubFldReqId")


go
```

```
/* Create new table "AssGroup".
                */
/* "AssGroup" : Assessment group can be used to group together individual
assignments for a particular student. These can then be used to match
against the qualification requirements. For example, an assessment group
could be created to cluster all of the student previous qualifications and
unit standards to determine which requirements of their current
qualification goal are met by them. Alternatively, a single assessment
group could be used for each student in which every unit standard they have
been accredited with is stored. */
/*      "AssGroupId" : Assessment Group is a group of Unit Standards
achievements for a particular Student. Assessment Groups are used to match
against a set of Qualification requirements to determine whether they have
been met. */
/*      "AssGroupName" : Assessment Group Name is a label for an Assessment
Group                */
create table "AssGroup" (
        "AssGroupId" int not null,
        "AssGroupName" char(30) not null)

go

alter table "AssGroup"
        add constraint "AssGroup_PK" primary key ("AssGroupId")


go

/* Add the remaining keys, constraints and indexes for the table "DomReq".
                */
create unique index "DomReq_AK1" on "DomReq" (
        "ElecId",
        "DomId")


go

alter table "DomReq" add constraint "DomReq_AK1_UC1" unique (
        "ElecId",
        "DomId")


go

/* Add the remaining keys, constraints and indexes for the table
"CompUnitStand".            */
create unique index "CompUnitStand_AK1" on "CompUnitStand" (
        "StudId",
        "UnitRef")


go

alter table "CompUnitStand" add constraint "CompUnitStand_AK1_UC1" unique (
        "StudId",
        "UnitRef")


go

/* Add the remaining keys, constraints and indexes for the table
"CompQual".                 */
create unique index "CompQual_AK1" on "CompQual" (
        "StudId",
        "QualRef")


go

alter table "CompQual" add constraint "CompQual_AK1_UC1" unique (
        "StudId",
```

```
        "QualRef")

go

/* Add the remaining keys, constraints and indexes for the table
"SubFieldReq".                  */
create unique index "SubFieldReq_AK1" on "SubFieldReq" (
        "ElectiveId",
        "SubFieldId")


go

/* Add the remaining keys, constraints and indexes for the table
"Subfield".                     */
create unique index "Subfield_AK1" on "Subfield" (
        "SubfieldName")


go

alter table "Subfield" add constraint "Subfield_AK1_UC1" unique (
        "SubfieldName")


go

/* Add the remaining keys, constraints and indexes for the table
"Qualification".                */
create index "SubField_Qualification_FK1" on "Qualification" (
        "SubFieldId")


go

create unique index "Qualification_AK2" on "Qualification" (
        "Title")


go

alter table "Qualification" add constraint "Qualification_AK2_UC1" unique (
        "Title")


go

/* Add the remaining keys, constraints and indexes for the table "Domain".
                */
create unique index "Domain_AK1" on "Domain" (
        "DomainName")


go

alter table "Domain" add constraint "Domain_AK1_UC1" unique (
        "DomainName")


go

/* Add the remaining keys, constraints and indexes for the table "Unit
Standard".               */
create unique index "Unit Standard_AK1" on "Unit Standard" (
        "UnitTitle")


go

alter table "Unit Standard" add constraint "Unit Standard_AK1_UC1" unique (
```

```
        "UnitTitle")

go

/* Add the remaining keys, constraints and indexes for the table "Student".
                */
create unique index "Student_AK1" on "Student" (
        "NZQARef")


go

alter table "Student" add constraint "Student_AK1_UC1" unique (
        "NZQARef")


go

/* Add the remaining keys, constraints and indexes for the table "Field".
                */
create unique index "Field_AK1" on "Field" (
        "FieldName")


go

alter table "Field" add constraint "Field_AK1_UC1" unique (
        "FieldName")


go

/* Add the remaining keys, constraints and indexes for the table
"Achievement".                  */
create unique index "Achievement_AK1" on "Achievement" (
        "StudId",
        "UnitStandardRef")


go

alter table "Achievement" add constraint "Achievement_AK1_UC1" unique (
        "StudId",
        "UnitStandardRef")


go

/* Add the remaining keys, constraints and indexes for the table
"Programme".                    */
create unique index "Programme_AK1" on "Programme" (
        "ProgName")


go

alter table "Programme" add constraint "Programme_AK1_UC1" unique (
        "ProgName")


go

/* Add the remaining keys, constraints and indexes for the table
"UnitStandardReq".              */
create unique index "UnitStandardReq_AK1" on "UnitStandardReq" (
        "ElectiveId",
        "UnitStandardRef")


go
```

```
/* Add foreign key constraints to table "Withdrawal".
                */
alter table "Withdrawal"
        add constraint "Student_Withdrawal_FK1" foreign key (
                "StudId")
         references "Student" (
                "StudId")

go

alter table "Withdrawal"
        add constraint "Programme_Withdrawal_FK1" foreign key (
                "ProgId")
         references "Programme" (
                "ProgId")

go

/* Add foreign key constraints to table "Enrolment".
                */
alter table "Enrolment"
        add constraint "Student_Enrolment_FK1" foreign key (
                "StudId")
         references "Student" (
                "StudId")

go

alter table "Enrolment"
        add constraint "Programme_Enrolment_FK1" foreign key (
                "ProgId")
         references "Programme" (
                "ProgId")

go

/* Add foreign key constraints to table "DomReq".
                */
alter table "DomReq"
        add constraint "Elective_DomReq_FK1" foreign key (
                "ElecId")
         references "Elective" (
                "ElectiveId")

go

alter table "DomReq"
        add constraint "Domain_DomReq_FK1" foreign key (
                "DomId")
         references "Domain" (
                "DomainId")

go

/* Add foreign key constraints to table "CompUnitStand".
                */
alter table "CompUnitStand"
        add constraint "Student_CompUnitStand_FK1" foreign key (
                "StudId")
         references "Student" (
                "StudId")

go

alter table "CompUnitStand"
        add constraint "Unit Standard_CompUnitStand_FK1" foreign key (
                "UnitRef")
         references "Unit Standard" (
                "UnitStandardRef")
```

```
go

alter table "CompUnitStand"
        add constraint "AssGroup_CompUnitStand_FK1" foreign key (
                "AssGroupId")
         references "AssGroup" (
                "AssGroupId")

go

/* Add foreign key constraints to table "CompQual".
                */
alter table "CompQual"
        add constraint "Student_CompQual_FK1" foreign key (
                "StudId")
         references "Student" (
                "StudId")

go

alter table "CompQual"
        add constraint "Qualification_CompQual_FK1" foreign key (
                "QualRef")
         references "Qualification" (
                "QualRefId")

go

alter table "CompQual"
        add constraint "AssGroup_CompQual_FK1" foreign key (
                "AssGroupId")
         references "AssGroup" (
                "AssGroupId")

go

/* Add foreign key constraints to table "QualField".
                */
alter table "QualField"
        add constraint "Qualification_Qual Field_FK1" foreign key (
                "QualRefId")
         references "Qualification" (
                "QualRefId")

go

alter table "QualField"
        add constraint "Field_QualificationField_FK1" foreign key (
                "FieldId")
         references "Field" (
                "FieldId")

go

/* Add foreign key constraints to table "SubFieldReq".
                */
alter table "SubFieldReq"
        add constraint "SubField_SubFieldReq_FK1" foreign key (
                "SubFieldId")
         references "Subfield" (
                "SubFieldId")

go

alter table "SubFieldReq"
        add constraint "Elective_SubFieldReq_FK1" foreign key (
                "ElectiveId")
         references "Elective" (
                "ElectiveId")

go
```

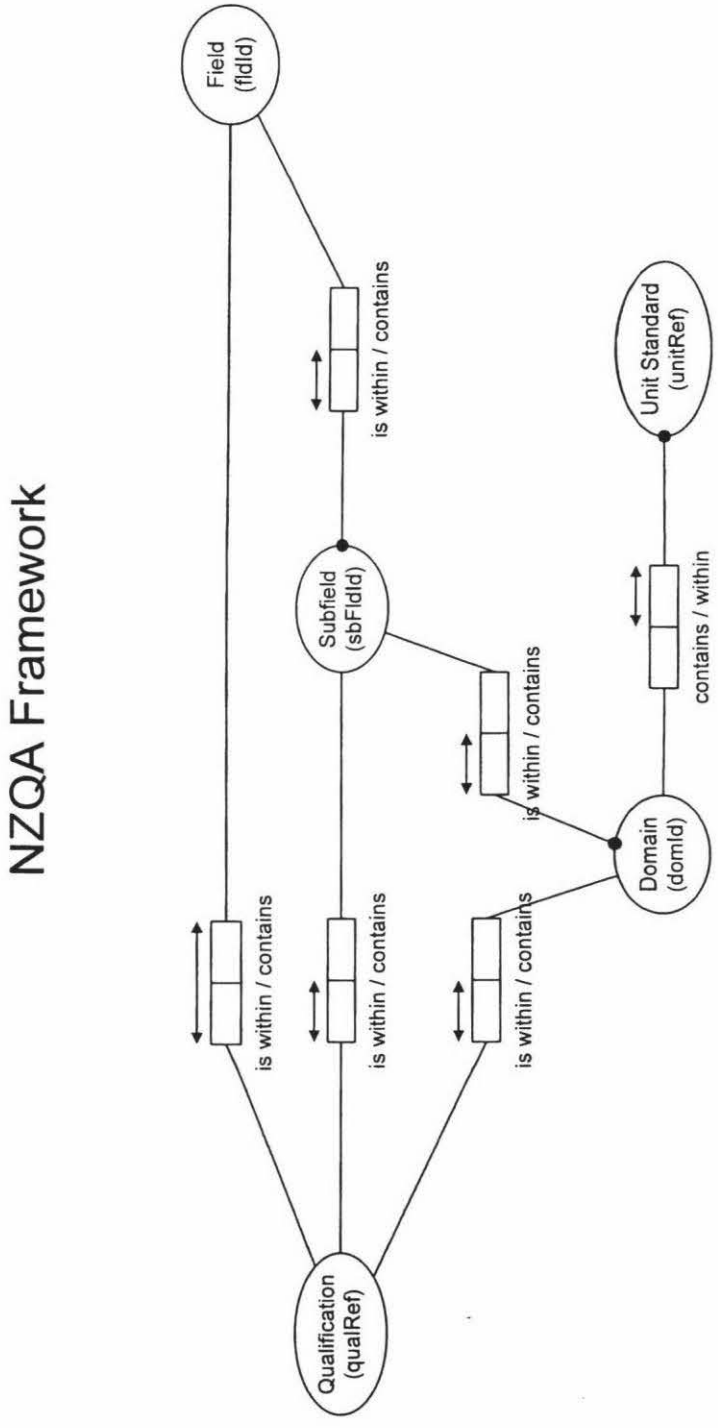
```
/* Add foreign key constraints to table "Subfield".
                */
alter table "Subfield"
        add constraint "Field_SubField_FK1" foreign key (
                "FieldId")
         references "Field" (
                "FieldId")

go

/* Add foreign key constraints to table "Qualification".
                */
alter table "Qualification"
        add constraint "SubField_Qualification_FK1" foreign key (
                "SubFieldId")
         references "Subfield" (
                "SubFieldId")

go

alter table "Qualification"
        add constraint "Domain_Qualification_FK1" foreign key (
                "DomainId")
         references "Domain" (
                "DomainId")

go

/* Add foreign key constraints to table "Domain".
                */
alter table "Domain"
        add constraint "SubField_Domain_FK1" foreign key (
                "SubFieldId")
         references "Subfield" (
                "SubFieldId")

go

/* Add foreign key constraints to table "MaxCreditValue".
                */
alter table "MaxCreditValue"
        add constraint "Elective_MaxCreditValue_FK1" foreign key (
                "ElectiveId")
         references "Elective" (
                "ElectiveId")

go

/* Add foreign key constraints to table "MinCreditValue".
                */
alter table "MinCreditValue"
        add constraint "Elective_MinCreditValue_FK1" foreign key (
                "ElectiveId")
         references "Elective" (
                "ElectiveId")

go

/* Add foreign key constraints to table "Unit Standard".
                */
alter table "Unit Standard"
        add constraint "Domain_UnitStandard_FK1" foreign key (
                "DomainId")
         references "Domain" (
                "DomainId")

go

/* Add foreign key constraints to table "Elective".
                */
```

```
alter table "Elective"
        add constraint "Qualification_Elective_FK1" foreign key (
                "QualId")
         references "Qualification" (
                "QualRefId")

go

/* Add foreign key constraints to table "Achievement".
                */
alter table "Achievement"
        add constraint "Student_Achievement_FK1" foreign key (
                "StudId")
         references "Student" (
                "StudId")

go

alter table "Achievement"
        add constraint "AssGroup_Achievement_FK1" foreign key (
                "AssGroupId")
         references "AssGroup" (
                "AssGroupId")

go

alter table "Achievement"
        add constraint "Unit Standard_Achievement_FK1" foreign key (
                "UnitStandardRef")
         references "Unit Standard" (
                "UnitStandardRef")

go

/* Add foreign key constraints to table "UnitStandardReq".
                */
alter table "UnitStandardReq"
        add constraint "Elective_UnitStandardReq_FK1" foreign key (
                "ElectiveId")
         references "Elective" (
                "ElectiveId")

go

alter table "UnitStandardReq"
        add constraint "UnitStandard_UnitStandardReq_FK1" foreign key (
                "UnitStandardRef")
         references "Unit Standard" (
                "UnitStandardRef")

go

/* Add foreign key constraints to table "DomainReqLevel".
                */
alter table "DomainReqLevel"
        add constraint "DomReq_DomainReqLevel_FK1" foreign key (
                "DomReqId")
         references "DomReq" (
                "DomReqId")

go

/* Add foreign key constraints to table "SubFieldReqLevel".
                */
alter table "SubFieldReqLevel"
        add constraint "SubFieldReq_SubFieldReqLevel_FK1" foreign key (
                "SubFieldReqId")
         references "SubFieldReq" (
                "SubFieldReqId")

go
```

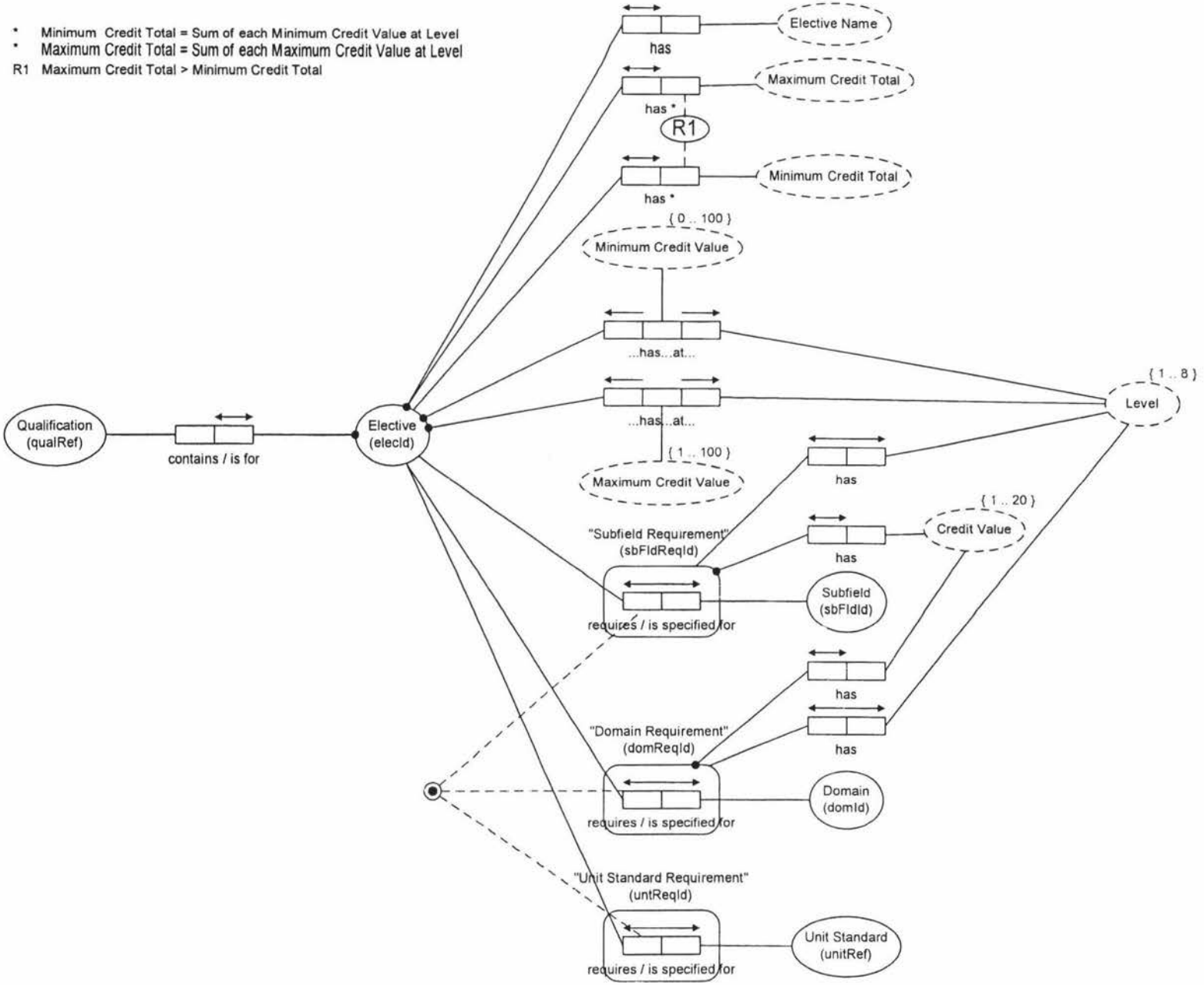
```
/* Add foreign key constraints to table "AssGrpUnitReq".
                  */
alter table "AssGrpUnitReq"
        add constraint "AssGroup_AssGrpUnitReq_FK1" foreign key (
                "AssGroupId")
         references "AssGroup" (
                "AssGroupId")

go

alter table "AssGrpUnitReq"
        add constraint "UnitStandardReq_AssGrpUnitReq_FK1" foreign key (
                "UnitReqId")
         references "UnitStandardReq" (
                "UnitReqId")

go

/* Add foreign key constraints to table "AssGrpDomainReq".
                  */
alter table "AssGrpDomainReq"
        add constraint "AssGroup_AssGrpDomainReq_FK1" foreign key (
                "AssGroupId")
         references "AssGroup" (
                "AssGroupId")

go

alter table "AssGrpDomainReq"
        add constraint "DomReq_AssGrpDomainReq_FK1" foreign key (
                "DomReqId")
         references "DomReq" (
                "DomReqId")

go

/* Add foreign key constraints to table "AssGrpSubFieldReq".
                  */
alter table "AssGrpSubFieldReq"
        add constraint "AssGroup_AssGrpSubFieldReq_FK1" foreign key (
                "AssGroupId")
         references "AssGroup" (
                "AssGroupId")

go

alter table "AssGrpSubFieldReq"
        add constraint "SubFieldReq_AssGrpSubFieldReq_FK1" foreign key (
                "SubFieldReqId")
         references "SubFieldReq" (
                "SubFieldReqId")

go

/* Create procedure/function DomReq_MR1.
                  */
/* /* The constraint:
                     */ */
/* /* Each Elective requires some Domain or requires some Subfield or
requires some Unit Standard. */ */
/* /* is enforced by the following DDL.
                     */ */
Create Procedure sp_DomReq_MR1 as
/*    Microsoft Visual Studio generated procedure code. */
if (
    not exists (select * from "DomReq" where
                "DomId" is null and
                "SubFieldId" is null and
                "UnitStandardRef" is null and
```

```
            )
)
  return 1
else
  return 2
/* End sp_DomReq_MR1
                */



go

/* Create procedure/function Student_enrols_Programme_Date_equal2.
                */
/* /* The constraint:
                */ */
/* /* Student s withdraws from Programme p on some Date if and only if
                */ */
/* /*   Student s enrols on Programme p on some Date.
                */ */
/* /* is enforced by the following DDL.
                */ */
Create Procedure sp_Student_enrols_Programme_Date_equal2 as
/*   Microsoft Visual Studio generated procedure code. */
if (
    not exists (select * from "Withdrawal" X where
                not exists (select * from "Enrolment" Y
                            where X.StudId = Y.StudId and X.ProgId" =
Y.ProgId)) and
    not exists (select * from "Enrolment" X where
                not exists (select * from "Withdrawal" Y
                            where Y.StudId = X.StudId and Y.ProgId =
X.ProgId))
)
  return 1
else
  return 2
/* End sp_Student_enrols_Programme_Date_equal2
                */



go

/* This is the end of the Microsoft Visual Studio generated SQL DDL script.
                */
```

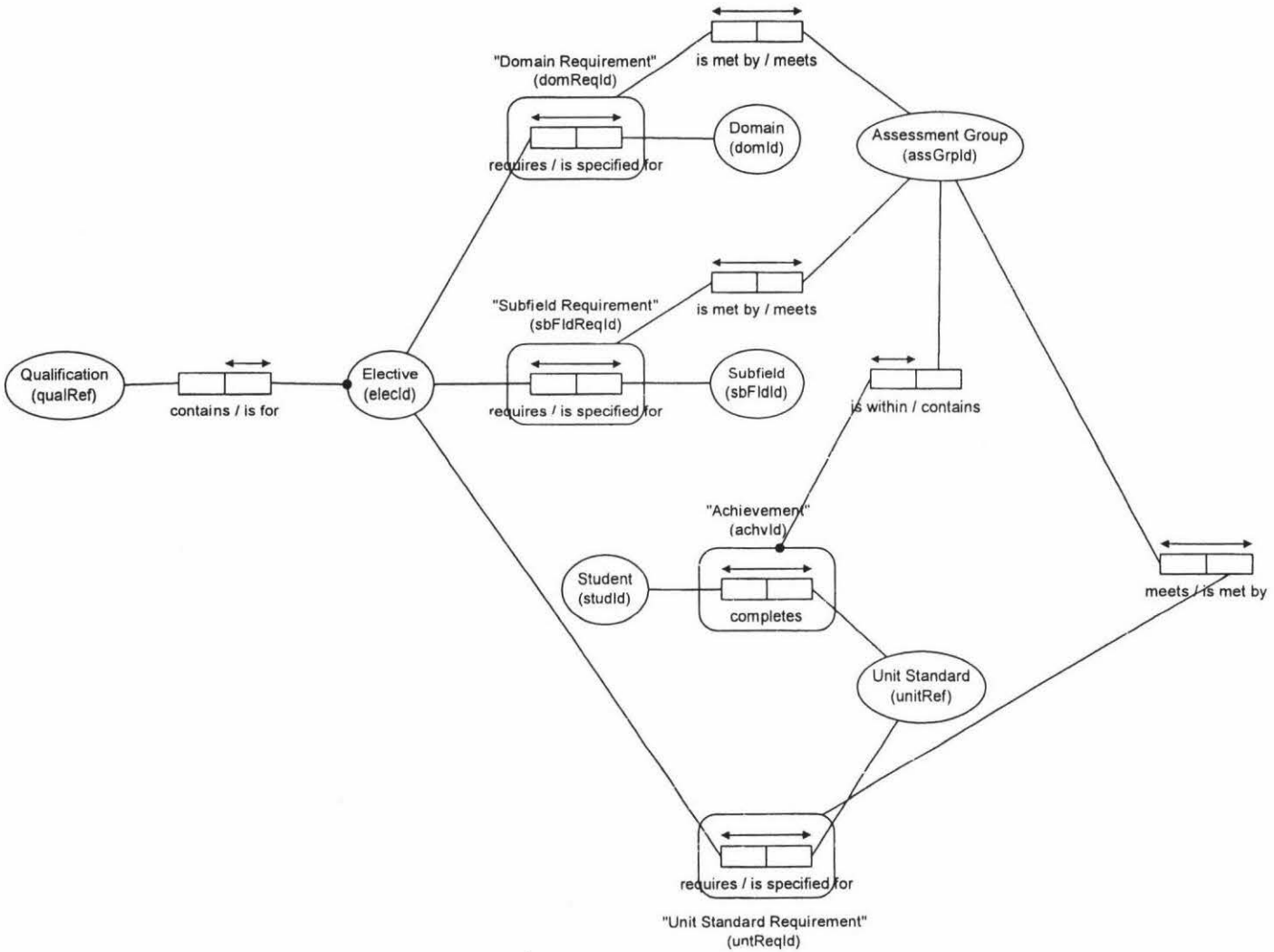
Appendix 9 - ORM Models for Student Mentoring System



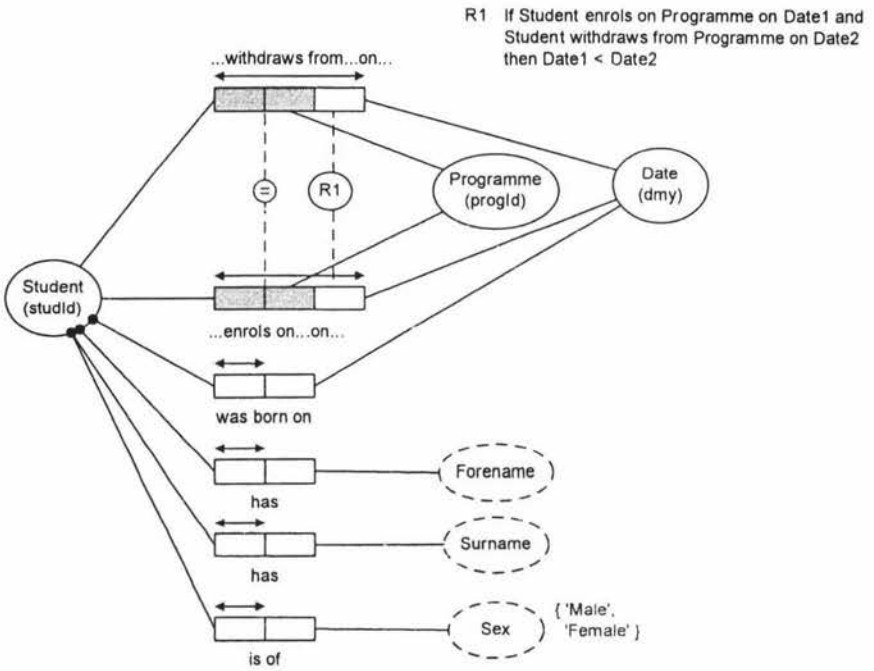
Elective


* Minimum Credit Total = Sum of each Minimum Credit Value at Level
* Maximum Credit Total = Sum of each Maximum Credit Value at Level
R1 Maximum Credit Total > Minimum Credit Total
Elective Name
has
Maximum Credit Total
has *
R1
Minimum Credit Total
has *
{ 0 .. 100 }
Minimum Credit Value
...has...at...
...has...at...
{ 1 .. 8 }
Level
Qualification (qualRef)
contains / is for
Elective (elecId)
{ 1 .. 100 }
Maximum Credit Value
has
{ 1 .. 20 }
Credit Value
"Subfield Requirement" (sbFldReqId)
has
Subfield (sbFldId)
requires / is specified for
has
has
"Domain Requirement" (domReqId)
Domain (domId)
requires / is specified for
"Unit Standard Requirement" (untReqId)
Unit Standard (unitRef)
requires / is specified for

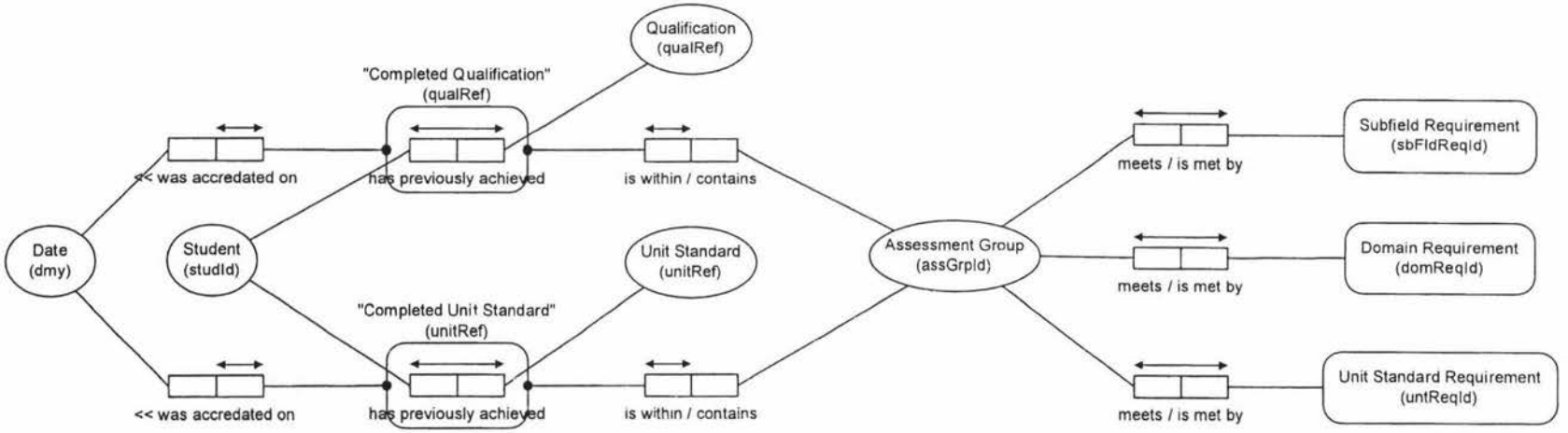
Student Achievement



Student


R1 If Student enrols on Programme on Date1 and Student withdraws from Programme on Date2 then Date1 < Date2
...withdraws from...on...
=
R1
Programme (progId)
Date (dmy)
Student (studId)
...enrols on...on...
was born on
Forename
has
Surname
has
Sex
{ 'Male', 'Female' }
is of

Credit Transfer



Appendix 10 – Business Rule Model
Reproduced from GUIDE – Business Rules Project, 1997