

Copyright is owned by the Author of the thesis. Permission is given for a copy to be downloaded by an individual for the purpose of research and private study only. The thesis may not be reproduced elsewhere without the permission of the Author.

**INTECoM:**  
**An integrated conceptual data modelling framework.**

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

**Doctor of Philosophy**

in

**Information Systems**

**at Massey University, Palmerston North**

**New Zealand**

**Clare Frances Atkins**

**2000**

## Errata Sheet

Page	Location	Action	Content
6	Para 3 - line 2	Insert Footnote after word NIAM	Originally standing for Nijssen's Information Analysis Method, it has been renamed at Nijssen's own request and this is now the <b>accepted terminology</b> (e.g. Sharp,1994)
121	Para 2	Replace last sentence with	Such an attempt is the subject of the remainder of this study which also includes in Chapter 12, the construction of an appropriate quality framework based on the issues discussed here.
175	Figure 20	Delete / Replace with	Diagram below
179	Para 1 line 1	Delete sentence 2	
180	Section 2.4.3.1 List point 3	Delete/ Replace with	double headed arrow crow'sfoot
180	Section 2.4.3.1 List point 4	Delete	or arrowhead
180	Section 2.4.3.1 List point 5	Delete/ Replace with	arrowheads barred crow'sfeet
185	Para 3 line 6	Delete/ Replace with	in practice in this development
221	End of para 1	Insert	The development of the INTECoM framework has also raised a number of interesting issues for research, which are discussed in more detail from <a href="#">page 236</a> .
229	Para 2 line 3	Delete	prescriptive
284	Fact Type 6 Example	Delete last 3 sentences	

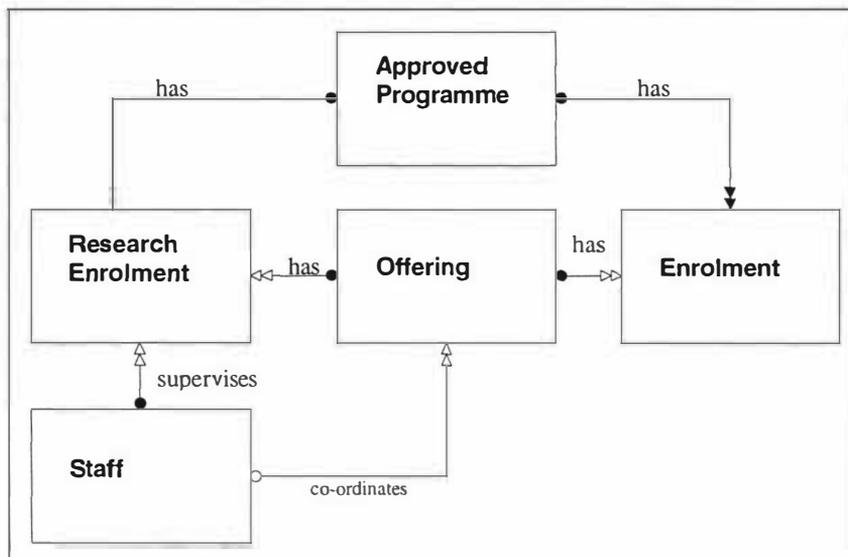


Figure 20 One solution to the Research Supervisor Problem

# Abstract

Conceptual data models, a fundamental component of information systems development, traditionally play two essential roles, as communication tools and database design blueprints. However, despite their importance to the success of information systems, and a considerable amount of research effort, no definitive method for constructing them has yet been described. Entity-Relationship (E-R) Modelling, accepted as a *de facto* standard for a number of years, has been increasingly criticised. A number of alternatives have been proposed and some, such as Object-Oriented modelling, have gradually been accepted by the practitioner community. Nevertheless, effective conceptual data modelling continues to be recognised as a difficult activity, both to teach and to practice.

This study investigates conceptual data modelling in the context of the relational database development process. However, rather than specifying a new method or exploring the efficacy of existing ones, it focuses on the nature of the activity itself. The construction of a conceptual data model encompasses both the analysis and design stages of systems development. Some fundamental differences in modeller behaviour, required by these activities, are explored. The disparate purposes of a conceptual data model are also investigated and the effectiveness of using the same modelling method in both stages, and for both purposes, is questioned. To explore the suitability of current methods to specific development activities, the procedures inherent in the use of two conceptual data modelling approaches, E-R Modelling and the Natural Language Information Analysis Method (NIAM) are also investigated.

The result of this exploration is the recognition that the methods have exclusive strengths and that it is more productive to view them as complementary rather than competing. Consequently, a database design framework, INTECoM, is constructed in which the two methods are integrated and matched to the activities for which they are most suited. The framework is supplemented by a new technique, NaLER, to facilitate communication in the design stage. The soundness and viability of this theoretical framework is examined through its use on a small development and the implications of the adoption of INTECoM on both education and practice are considered.

## Preface

*“The inherent structure of the E-R approach can be easily explained using natural language. For this reason, the 5 -10 year time frame will bring the combining of the E-R approach with the linguistic approach. Future modeling scenarios will involve a natural language interface, where the E-R model underlies with an interpreter and is invisible to the user. (Hawryszkiewicz, 1987 p.471)*

Hirschheim *et al.* (1995) comment on the large amount of research that has been amassed in the fields of “information systems development, in general, and data modelling in particular”. They continue,

“A wealth of research in these fields has produced an astonishing array of empirical results and practical insights, conceptual and terminological diversity and confusion and a large suite of tools and methods. But as many researchers and practitioners alike feel, these form an isolated, disjoint and often contradictory amalgam of knowledge” (p.xi).

This researcher concurs with this view and acknowledges their contribution in highlighting the diversity and confusion, which undoubtedly exists. While this study does not include significant discussion of their ideas, their informed analysis and incisive comments underlie many of its observations. Via, a close analysis of the literature and a synthesis of existing knowledge within the area of “fact-based data modelling” (*ibid.* p.28), this study attempts to “to shed light on similarities where they exist and to discuss possible directions for improvement” (*ibid.* p.xi).

This research is motivated by the belief that the nature of the conceptual data model has changed fundamentally over the last twenty years and that both the nature of these changes, and the purpose of the conceptual model, are not always clearly understood. Practitioners and academics alike, provide many independent definitions relevant to their specific viewpoint or purpose, with few attempts to provide a holistic description. Indeed, the analogy of six blind individuals attempting to construct an accurate image of an elephant by discretely describing its constituent parts may well be appropriate (Pletch, 1989). Additionally, the variety of research methods that has been, and continues to be, used, result in research findings that are often perplexingly

contradictory, untimely, and notable for their inability to inform practice. Rather than adding more experimental work to this *mélange*, this study focuses on, and attempts to clarify, some of the issues arising from this fragmented understanding. The principles underlying the conceptual data modelling process are exposed and existing tools, techniques and methods are investigated. The purpose of this critical examination is to explore whether integration can improve both the process of conceptual data modelling and the appropriateness and quality of the outputs.

The philosophical basis for this research is thus broadly interpretivist. Interpretivism, which has recently "...emerged as an important strand in information systems research" (Klein & Myers, 1998 p.2), has been adopted by a number of IS researchers (e.g. Walsham, 1993; Boland, 1991). Klein and Myers (1998) comment that,

"IS research can be classified as interpretive if it is assumed that our knowledge of reality is gained only through social constructions such as language, consciousness, shared meanings, documents, tools and other artifacts. Interpretive research does not predefine dependant and independent variables but focuses on the complexity of human sense making"(p.5)

Likewise, Walsham (1993) describes interpretivism as "an epistemological position concerned with approaches to the understanding of reality and asserting that all such knowledge is necessarily a social construction and thus subjective" (p.5). In discussing the use of interpretive methods for case study research, he later comments on the interrelationship between epistemology and research methods,

"If one adopts a positivist epistemological stance, then statistical generalisability is the key goal. However, from an interpretivist position, the validity of an extrapolation from an individual case or cases depends not on the representativeness of such cases in a statistical sense, but on the plausibility and cogency of the logical reasoning used in describing the results from the cases, and in drawing conclusions from them" (*ibid.* p.15)

Although this study does not involve the construction of case studies, it does rely on the plausibility and cogency of its logical reasoning to make sense of a particular set of IS development activities. It makes no claim to provide empirical evidence of the validity of the conclusions that are drawn but lays down threads of discrete arguments that are synthesised into a credible scenario. In so doing, this researcher has been influenced by, and taken cognisance of the arguments of researchers such as Lyytinen and Klein (1985), Galliers (1993), Ciborra (1997), Walsham (1995), and Klein and Myers (1999).

Ciborra (1997), in particular, has identified a crisis in IS research which he suggests stems from the general adoption of the “paradigm of the natural sciences and the relevant methodologies of measurements, normalisation and calculation.”(p.1551). He continues,

“What I am concerned with here is something subtly pervasive: it is for example, that in order to show that structured methodologies are a failure or plainly not used, one has to adopt a structured scientific method to empirically measure the phenomenon, in order to be credible and legitimate; and even then, being methodologies at the core of our discipline, these empirically measured facts still get to be dismissed” (*ibid.* p.1552).

This, he argues, provides a disservice to the IS community as “we tend to forget the role of human choice behind the technical artefacts, and study the user side of information systems by adopting the methods of the natural sciences” (*ibid.* p.1552). In response to this tendency, some researchers have looked instead to the social science disciplines for more appropriate methods (e.g. Walsham 1995; Klein & Myers, 1999). Galliers (1993) has suggested that the mode of subjective/argumentative research, which is the primary form adopted in this thesis, is appropriate for investigating methodological issues and categorises it as a post-positivist, interpretive approach.

Most of the more recent research effort, in the conceptual data modelling area, has concentrated on positivist, empirical studies that either compare and refine various qualities of different modelling formalisms or observe the behaviour exhibited by modellers with differing levels of expertise. However, perhaps because of the difficulty in designing pertinent experiments, little attention has been paid to the “*ways of working*” (Bronts *et al.*, 1995 p.214) inherent in the different formalisms. No studies were discovered that investigated whether the different behaviour required of the modeller by different formalisms had any effect on the quality of the final product. Likewise, there has been little, if any, consideration given to the data modelling requirements of the different stages of the information systems life cycle. Consequently, the appropriateness, of different techniques for the various tasks required by these stages, does not appear to have been adequately examined either.

This study then, is not interested in attempting to re-assess whether one formalism is more expressive, useable or comprehensible than another. Instead, it explores the wide range of functions ascribed to the conceptual data model, and focuses on the different

working practices and behaviours inherent in different conceptual data modelling formalisms. It then investigates the different 'ways of working' required by the activities of analysis and design and explores the question of whether some formalisms are inherently more suited to certain stages of information systems development than others.

In order to establish a base from which a useful integration can be derived it is necessary to also explore a number of related areas. Therefore, this study also examines various issues including; the history of conceptual data modelling; pedagogical issues; and methods that have previously been used to evaluate the quality and effectiveness of conceptual data models. A more detailed look at the processes involved in developing a conceptual data model with both Entity-Relationship (E-R) and NIAM (Natural Language Information Analysis Method) techniques is also undertaken. The end result of this critical and hermeneutic examination is the delineation of a framework within which these particular techniques can be used to greater advantage than the use of either of them alone.

The apparent contradiction in investigating the use of data modelling tools, classified as objectivist by Klein and Hirschheim (1987), to undertake an activity, data modelling, classified by them as subjectivist, is recognised. However, this study focuses on current education and practice and it is useful to accept and work within this apparent paradox. As Weber (1997) points out the ontological assumptions underlying the data modelling activity comes from the modellers themselves rather than from the data modelling grammars they choose to use. This study also broadly concurs with Kent's (1978) observation that,

“[A]t bottom we come to this duality. In an absolute sense there is no singular objective reality. But we can share a common enough view of it for most of our working purposes so that reality does appear to be objective and stable” (p.203).

Following a general introduction, and a working definition of some of the more overloaded terms in the first two chapters, Chapter 3 provides a historical perspective on the development of the data modelling activity. In so doing, the chapter explores the alteration in the purposes and use of a conceptual data model and raises some fundamental questions regarding the activity itself. Chapter 4 explores these issues in

more detail and investigates their implications on both practice and education. The process of constructing an E-R Model is discussed in Chapter 5 and the predominance of the Chen (1976) E-R Model, widely assumed by many academic writers, is challenged. In particular, the descriptive, creative nature of the process of E-R modelling and its appropriateness for analytical activity is explored. An alternative method, NIAM Conceptual Schema Design Procedure (NIAM-CSDP), is described in Chapter 6, with a continuing investigative emphasis on the procedure by which a model is created.

Having thus established a basis for such a discussion, Chapter 7 provides a comparison of the 'ways of working' of the two techniques. There is no attempt to establish any superiority of one method over another. Instead, the emphasis is on matching the different approaches to appropriate stages of the information systems development life cycle. In Chapter 8 the perspective of the analysis shifts slightly, to investigate the means by which the quality of conceptual models has been evaluated. This investigation confirms the existence of diverse definitions of a conceptual data model and highlights a number of issues that spring from this lack of consensus. It also shows that much of the previous research has sought to establish one modelling facility as the 'best', to the exclusion of all others. This study takes the position that the search for a 'holy grail' of data modelling, is inappropriate, wasteful and dangerous and has diverted attention away from both an investigation of the nature of the activities required by database development and on the construction of tools to match those needs.

Consequently, Chapter 9 proposes a framework, INTECoM in which elements of the NIAM and E-R approaches are matched to appropriate needs and used productively together. Chapter 10 provides a new technique, NaLER for extracting NIAM-like sentences from E-R models. It justifies the need for such a technique primarily as a means of constructing a formal, natural language view of the final design model and facilitating the audit of the conceptual data modelling process, but also to assist in model interpretation for both non-technical users and inexperienced modellers.

A small development to demonstrate the viability of the INTECoM framework was undertaken and is described in Chapter 11, while the issues of quality evaluation related to the use of the framework are discussed in Chapter 12. The final version of INTECoM

is defined in Chapter 13 and the study concludes, in Chapters 14 and 15, by looking at the implications inherent in the adoption of the INTECoM framework, for both the education of future data modellers and the practice of data modelling. The implications of using the NaLER technique is also discussed and some potential criticisms are addressed. A number of limitations of this work are highlighted and the possibilities for future research are delineated.

This thesis then, focuses on theory building rather than theory testing, constructing a new theory, in the form of an integrated framework, in which each element is justified by reference to, or inference from, the relevant literature. By taking a holistic view of the development of the data modelling process, it exposes and explores questions that have been largely overlooked by previous researchers. A potential criticism of this work could be that no empirical studies have been undertaken to assess the efficacy of the proposed framework. Apart from the concentration on theory building, there are several reasons for this. Firstly, the primary techniques within the framework, ER Modelling and Object-Role Modelling, are already well tested, used in the database community and considered to be effective. Secondly, the nature of the proposed framework is such that for a case study to hold any real significance it would need to be tested on a medium to large 'real-world' project. In addition, while such a test would undoubtedly be profitable, the scale of such a test puts it beyond the scope of this research. Instead, a small development undertaken by the researcher is described in detail, as a worked example, to demonstrate the overall structure of the framework, the inter-relationships that exist between the various elements and how the framework can be successfully instantiated.

There is one very clear and deliberate omission from this work and that is a detailed consideration of the object-oriented (O-O) approach to system development. While this omission is intentional and stems from a number of factors, O-O techniques have not been ignored. However, while the use of object-oriented techniques is undoubtedly increasing, as yet no standard process, whether *de facto* or actual, exists. The Unified Modelling Language (UML), currently supported commercially by Rational Software Corporation, brings together the major O-O methods that have been developed over the last ten years Booch (Booch, 1991, 1995), Object Modelling Technique (OMT) (Blaha *et al.* 1988; Rumbaugh *et al.*, 1991; Blaha & Premerlani, 1997) and Objectory (Jacobsen

*et al.*, 1992). UML provides a standard notation and development approach by bringing together the design strengths of Booch, the analysis strengths of OMT and the strong behavioural analysis of use cases (Quatrani, 1998). However, while the eventual acceptance of UML, as a *de facto* standard is probable, it is not yet recognised as such.

A more important consideration is that neither UML nor its predecessors provide a proven means of designing relational data structures. Textbooks describing the object-oriented approach to system development tend to emphasise the process and software aspects of development and either ignore the need for database design (e.g. Quatrani, 1998) or treat it as a required but additional technique (e.g. Larman, 1998). Indeed, as Quatrani (1998) states, “the Rational Objectory Process [*is*] an extensive set of guidelines that addresses the technical and organisational aspects of **software**<sup>1</sup> development” (p.8). The issues considered here are very specifically related to the development of relational databases and the detailed exploration of methods that do not seek to provide a means to design such structures is thus not relevant. Of all the O-O methods, OMT had placed the greatest emphasis on relational database design and the general approach to system development recommended in both UML and OMT is considered in Chapter 5. However, it is argued that while both the notation and the working language of these formalisms may differ, in terms of their approach to data identification and structuring, their ‘way or working’<sup>2</sup> is essentially no different from the approach required by traditional E-R Modelling. Indeed, Eaglestone and Ridley (1998) specifically state,

“ Data analysis methods provide systematic processes by which conceptual models are derived. One such method, more commonly associated with relational database design, is entity-relationship analysis [Chen 76]. This is not inherently linked to relational databases and is in fact the sort of process we would wish to undertake to design an object database” (p.276).

This research takes the view that the requirement to build sound data structures is likely to continue to be important for some time and that the need to formally analyse and record user data needs and to then transform them to effective relational databases, will thus remain. While the specific techniques that are used to meet these requirements may

---

<sup>1</sup> Emphasis added

<sup>2</sup> The meaning and importance of a method’s ‘way of working’ is considered fully in Chapter 5.

well change, the fundamental principles that underlie them are likely to remain constant. It is thus the adoption of the principles of the INTECoM framework that is advocated by this thesis rather than any specific data modelling formalisms with which it is instantiated.

## Acknowledgements

My initial thanks go to my ex-colleagues at the Statistics Division of the Inland Revenue in the UK. In particular, Dr Ron James who first inspired me to a deep interest in data modelling and Dave Boutwood, who not only taught me the value of constructive argument but also gave me plenty of practice.

For more specific help, I must thank my two supervisors at Massey University: Dr Daniela Mehandjiska-Stavreva, now of Bond University, Australia for her support and encouragement in bleak times and Professor Jon Patrick, now of the University of Sydney, for the challenging intellectual debate he provided and particularly for his assistance in formally defining the NaLER language.

In addition, Wen van Kersbergen, from the Amsterdam School of Business, gave me his invaluable guidance on the practical use of the NIAM-CSDP and InfoModeler™ and Dr Steve Hitchman provided me with many useful insights and criticisms, from both his academic and professional experience.

All my colleagues in the Department of Information Systems, particularly Chris Freyberg, Mike Ryder and Peter Blakey have given me both practical and moral support. Lastly, my heartfelt thanks to all the students on whom I have tested out my theories and explored my ideas; for their feedback, their enthusiastic participation and their patience.

## Publications

The following refereed papers have been directly based on work in this thesis.

Atkins, C.F.(1996): Prescription or Description: Some Observations on the Conceptual Modelling Process, in PURVIS, M.(ed.), *Proceedings of Software Engineering: Education and Practice Conference*, Dunedin, New Zealand, January: 34-41.

Atkins, C.F. and Patrick, J.D.(1998): NaLER: A Natural Language Method for Interpreting E-R Models, in PURVIS, M. (ed.), *Proceedings of Software Engineering: Education and Practice Conference*, Dunedin New Zealand, January: 2-9.

Atkins, C.F. and Patrick, J.D.(2000): NaLER: A Natural Language Method for Interpreting E-R Models, *Campus Wide Information Systems*, (forthcoming).



# Contents

Preface.....	v
Acknowledgements.....	xiii
Publications.....	xiii
1 Introduction .....	5
2 Clearing the Confusion.....	11
3 Data Modelling.....	25
4 Conceptual Data Modelling: some underlying issues .....	39
5 E-R Modelling: observations.....	57
6 NIAM: observations .....	71
7 E-R and NIAM: a comparison of approach .....	83
8 Evaluating Data Models .....	103
9 INTECoM: an integrated framework.....	123
10 NaLER: completing the circle .....	143
11 INTECoM: in practice.....	157
12 INTECoM: quality matters .....	189
13 INTECoM: an instantiation .....	205
14 Implications .....	221
15 Conclusion .....	233
References .....	239
Glossary.....	257
Appendices .....	263
Appendix 1 - InfoModeler™ transformations.....	265
Appendix 2 - NaLER Definition Language.....	267
Appendix 3 - ISPG Context Diagram.....	271
Appendix 4 - Analysis Documentation .....	273
Appendix 5 - Initial Design .....	295
Appendix 6 - Design in Progress.....	301
Appendix 7 - Design Verification .....	305
Appendix 8 - Equivalence Tables .....	315
Appendix 9 - Design Documentation.....	321
Appendix 10 - Verified Design Model.....	325
Appendix 11 - Design Innovation .....	329
Appendix 12 - Task Checklists .....	331

# List of Figures

Figure 1 ANSI/X3/SPARC Architecture - adapted from Avison (1992).....	26
Figure 2 A revised view of the ANSI 3 level architecture .....	32
Figure 3 Meta-Model Architecture.....	35
Figure 4 IS Development as a duality (de Carteret and Vidgen, 1995).....	45
Figure 5 Conceptual Modelling Activity Kim and March (1995) adapted. ..	54
Figure 6 The 7 steps of the NIAM-CSDP (Halpin, 1995).....	73
Figure 7 A set of example sentences for two qualified fact types .....	74
Figure 8 A simple NIAM diagram .....	75
Figure 9 A framework for IS methodologies, (Bronts et al, 1995) .....	85
Figure 10 Example of 'verbalization' report' from InfoModeler™.....	96
Figure 11 Natural language interpretation of E-R/R constructs.....	97
Figure 12 Lindland et al.'s (1994) Framework .....	112
Figure 13 Concepts in the framework of Krogstie et. al. (1995).....	114
Figure 14 An integrated conceptual data modelling approach.....	133
Figure 15 Example of Feedback from SERFER (Batra and Sein,1994) .....	146
Figure 16 NaLER - An overview .....	148
Figure 17 NaLER sentences and examples. ....	156
Figure 18 ISPG System - Context Diagram .....	159
Figure 19 Second draft design model.....	174
Figure 20 One solution to the Research Supervisor Problem .....	175
Figure 21 Final draft design model .....	177
Figure 22 Pre-Verification Design Model.....	180
Figure 23 INTECoM - Quality Framework.....	194
Figure 24 INTECoM - Instantiated Quality Framework .....	202
Figure 25 INTECoM Framework – Final Version.....	206
Figure 26 Create Analysis Model - Activities.....	208
Figure 27 Construct Design Model - Acitivities .....	212

# List of Tables

Table 1	Some definitions of conceptual modelling.....	39
Table 2	Evaluation criteria used in conceptual modelling studies.....	104
Table 3	Definitions of ‘conceptual model ‘ in comparative studies.....	110
Table 4	Approaches to quality in conceptual modelling.....	111
Table 5	Goals and Metrics of proposed quality frameworks.....	117
Table 6	Combined list of all entities from analysis model.....	166
Table 7	List of all entities sorted by their primary key attributes.....	167
Table 8	List of entities after initial merging.....	171
Table 9	Two-way sentences for the final draft design model.....	178
Table 10	Primary to foreign key links in final draft design model.....	178
Table 11	Initial quality criteria for INTECoM design model.....	190
Table 12	Quality evaluation of analysis model.....	196
Table 13	Analysis Task Checklist.....	197
Table 14	Quality evaluation of design model.....	200
Table 15	Design Task Checklist.....	201

# 1 Introduction

*"A beginning is the time for making sure that all the balances are correct" (Herbert, 1972).*

Analysing and describing the structure of data required by an organisation to support its information systems, is generally regarded as a fundamental activity in information system development (e.g. Hitchman, 1995; Kim & March, 1995; Nijssen & Halpin, 1989; Batra & Marakas, 1995; Hawryszkiewicz, 1997). This is particularly so where the system is, primarily, database focussed (Avison, 1992; Avison & Fitzgerald, 1995; Kroenke, 1992; Post, 1999). The growing recognition of the value of data to an organisation (Shanks & Darke, 1997) has also resulted in increasing importance being attached to its management and auditing (e.g. Amer, 1993; Kinn & Sullivan, 1994).

For these activities to be effective, an accurate, understandable, documented record of the consensus view of the organisation's data is required. It would, therefore, seem likely that the majority of organisations would produce some form of high level representation or model of their data and then utilise this information in the design of its information systems. However, related research, while not explicitly addressing this question, suggests that this may not be the case. In 1994 only 32% of surveyed organisations in the UK reported always using data analysis on projects (Hitchman, 1995). In the same year, in New Zealand, only 30% of surveyed CASE (Computer Aided Software<sup>1</sup> Engineering)/4GL (Fourth Generation Language) users reported the use of data analysis techniques (MacDonell, 1994). While this research does not, in itself, offer any substantive proof for the lack of high-level models, it does seem reasonable to suggest that organisations, which do not make use of such techniques at the project level, are unlikely to build an effective, corporate view of their data. Despite

---

<sup>1</sup> or Systems.

there being a number of perceived benefits to creating effective data models, (e.g. de Carteret & Vidgen, 1995; Simson, 1994), it would seem that almost 70% of organisations are choosing not to do so. It seems unlikely that organisations would choose to ignore obvious benefits, if the means of creating such useful models was straightforward, well understood and cost-effective. As Simson (1994 p.21) argues, "data modelling is not optional" for implicit in the creation of a database is the design of a data model. A database is by its very nature a data model and the only optional feature is the level of formality that has been followed in its construction. Organisations that choose not to undertake formal data modelling are, thus, not only unlikely to maximise their investment in their data but also in danger of creating poorly designed, undocumented databases.

There does not appear to be any obvious reason for this dichotomy, although Hitchman (1995) has suggested that there may be a lack of understanding of data models and that neither IS practitioners nor business clients find the current forms of data representations intuitive in use. Since its introduction in 1976, the E-R (Entity-Relationship) approach (Chen, 1976) or one of its many variations (e.g. Teorey *et al.*, 1986) has reportedly become the most ubiquitous technique for creating data representations (Batra *et al.*, 1990; Loosely & Gane, 1990; Flynn, 1998; Hawryskiewicz, 1997; Moody & Shanks, 1998). However, although a number of early studies commended the E-R model for its ease of use and intuitiveness (Brodie *et al.*, 1984; Teorey *et al.*, 1986; Yao *et al.*, 1982), the findings of the more recent work quoted above (Hitchman, 1995; MacDonell, 1994) would seem to undermine these views. Indeed, Moody and Osianlis (1996 p.506) writing from a practitioner's perspective argue that "while the Entity Relationship model is an excellent tool for designing database schemas, it never was and never will be suitable for direct consumption by business users." Additionally, some of the impetus behind the adoption of object-oriented techniques and the development of the Unified Modelling Language (UML) has been the desire to provide a more intuitive and natural view of reality (Martin and Odell, 1992).

Object-Role Modelling (ORM) (Biller & Neuhold, 1977) is a recognised alternative to the E-R approach. The technique is central to NIAM (Natural Language Information Analysis Method); a method based on a formalised natural language representation of

data. NIAM also lays out a specific process, the Conceptual Schema Design Procedure (CSDP), for the creation of data representations (Nijssen & Halpin, 1989; Halpin, 1995). Widely used in Australia and Europe, the NIAM-CSDP is significantly more prescriptive than any of the methods proposed for the creation of representations using traditional Entity-Relationship modelling techniques. Unlike E-R Modelling, it utilises a formal subset of natural language to determine 'facts' from which the O-R model is then built. The O-R is then transformed into a normalised relational data model.

These two approaches have been generally regarded as alternative methods to modelling and a number of studies have looked to compare their effectiveness both as representational tools and as useable methods (e.g. Shoval & Even-Chaime, 1987; Kim & March, 1995). None of the studies investigate the differences in the kind of cognitive and practical behaviour that is required of the modeller (Sutcliffe & Maiden, 1992), although one (Batra & Davies, 1992) does examine cognitive differences between expert and novice modellers. In addition, these studies do not generally make a distinction between the analysis and design activities that are an integral part of the data modelling process. Consequently they do not address the differing requirements of the different stages of the modelling process, the differing purposes that the resultant models are required to serve nor the kind of modelling behaviour that is likely to be most effective at which stage. Likewise there is no discussion on the suitability of the various methods to the various purposes of the model, despite these having been clearly set out in early research (Tschritzis & Lochovsky, 1982).

However, there are significant differences in the processes commonly used to construct either an E-R or an ORM conceptual model. This study seeks to explore these procedural and behavioural differences and highlight their major strengths, weaknesses and appropriateness to different aspects of the data modelling process. Therefore, how the processes are generally taught and used and the implications of this, on both the ease of learning and the reliability of the outputs, are also investigated. It is argued that the facilities offered by the two methods are complementary rather than competing and that each should be recognised and used in an appropriate way and for an appropriate purpose. The study suggests that elements of the NIAM-CSDP are best suited to the analysis stage of conceptual data modelling while the processes inherent in the creation an E-R model are better suited to design activities.

Bringing together the various strands mentioned above, this study first proposes and outlines a framework for data analysis and design that seeks to create a better match between the procedures and techniques within the two methods and the activities they are designed to support. This framework, termed INTECoM, (Integrated Conceptual Modelling) is based in current practice and facilitates the integration of techniques taken from both the NIAM-CSDP and E-R Modelling. In addition, a new technique is developed to provide a formal, natural language representation of an E-R Model. It does not suggest that these are the only techniques that can be used in this way, merely that they are appropriate and conveniently highlight several significant and pertinent issues. Thus, INTECoM does not claim to be a definitive conceptual data modelling method but a framework that can, potentially, be instantiated with a variety of techniques.

The proposed framework, instantiated with these techniques, is then used to undertake a small example development. The progress of this development is described in detail and several observations are recorded. This practical application of the framework, while confirming the validity of the underlying principles, necessitates the detailing of a number of strategies and quality measures. These are finally consolidated into a definitive description of the framework. Recognition, and acceptance, of the validity of the principles on which INTECoM is built, would have a profound effect on the practice of conceptual data modelling and, subsequently, on the education of future data modellers. Therefore, the study concludes by considering some of these implications.

Before attempting to delineate any consequential arguments however, a very close examination of some data modelling terminology is required. While almost any intellectual activity can be improved by beginning with a clear basis for communication, in this particular endeavour, such an undertaking is, not just desirable but essential. Hirschheim *et al.* (1995) remark on the “terminological diversity and confusion” (p.xi) that abounds in the data modelling community, despite Olle’s (1993) attempt to inject some sense of order. Clearly, a full and definitive explanation of all such terminology is not the focus of this research, which, has, to some extent, to work within this “disjoint and often contradictory amalgam of knowledge” (Hirschheim *et al.*, 1995, p.xi). Nevertheless there are a number of terms and concepts that are used in specific ways in the following discussions and which therefore require some careful consideration. In

addition, there are some interesting observations to be made through a close examination of the various meanings that have been ascribed to certain terms.

Consequently, the following chapter deals exclusively with some of the more contentious expressions. These are summarised in the glossary on page 257, which also contains working definitions of other words and terms as they are used in the context of this thesis. However, one term, 'conceptual data modelling', perhaps the most contentious of all, is dealt with separately in Chapter 4.



## 2 Clearing the Confusion

*“ ‘When I use a word, ‘ Humpty Dumpty said, in rather a scornful tone, ‘it means just what I choose it to mean - neither more nor less.’*

*‘The question is,’ said Alice, ‘whether you can make words mean so many different things.’  
‘The question is,’ said Humpty Dumpty, ‘which is to be master -- that’s all.’ “ (Lewis Carroll, 1871)*

### Introduction

Any discussion of the topics raised in this study immediately stumbles over the plethora of meanings, synonyms, homonyms and misunderstandings, employed to describe the fundamental elements of the subject area. Ironically, in a discipline concerned, primarily, with attempting to extract exact and consensual meanings from ambiguous and ill-defined natural language descriptions, there is no clearly defined vocabulary to describe the nature of the techniques and concepts that are employed. A number of terms are either contentious, overloaded with meaning, well defined but misused or simply not clearly defined. To increase the confusion, practitioners and academics have developed different terminologies and in any piece of writing it is not always clear which is being used, as both, or a combination, may be used in either context. Some of the more ubiquitous terms are discussed here, while others are investigated in detail in subsequent chapters, as they become relevant to the developing arguments.

### Data and Information

As early as 1966 clear definitions of ‘data’ and ‘information’, in respect of their use within a computing context, were provided as,

*“Data: A representation of facts or ideas in a formalised manner capable of being communicated by some process. Note: The representation may be more suitable either for human interpretation (e.g. printed text) or for interpretation by equipment (e.g. punched cards or electrical signals) (IFIP ICC, 1966, quoted in Olle, 1993)”*

and

“Information: In data processing, the meaning that a human ascribes to data by means of the known conventions used in its representation. Note: the term has a sense wider than that of ordinary information theory and nearer to that of common usage...” (*ibid.*)

However, early authoritative writers in this area such as Kent (1978) and Chen (1976) found no need to define data at all before discussing it. In the days of ‘data processing’, it was clear that data referred to small pieces of “information operated on by a computer program” as the Collins dictionary (Hanks, 1979) defined it in 1979. Despite the IFIP clarification quoted above, no clear distinction between data and information seemed generally necessary. Data, defined as “a series of observations, measurements or facts” in close association to its Latin root of *dare* - to give, was considered synonymous with information, drawn from the Latin root *informare* - to give form to (Hanks, 1979). In contradiction of its own definition, the Collins dictionary continues by saying that information is also “the results derived from the processing of data according to programmed instructions” (Hanks, 1979). Tsichritzis and Lochovsky (1982) were among the earliest to find it necessary to provide a distinction between these terms, and hence have provided the basis for most subsequent definitions. “Data correspond to discrete, recorded facts about phenomena or ideas”, that are “worth not only thinking about, but also worth recording in a somewhat precise manner”, while “information is an increment of knowledge that can be inferred from data” (Tsichritzis & Lochovsky, 1982 p.3).

However, more recent writers (e.g. Firms, 1990; Avison, 1992; Ricardo, 1990; Kroenke, 1992) also begin with definitions of these same terms, perhaps in recognition of the confusion that now appears to exist over their use in phrases such as data modelling and information modelling. In general, these later definitions do not succeed in significantly adding value to their earlier counterparts although Schenck & Wilson (1994), providing echoes of the early IFIP definition, are an exception. They make an explicit and useful connection between data as “symbols which represent information for processing purposes” (p.xxvii) and the need for agreement on the meaning of those symbols. They also bring a refreshing succinctness in quoting Mary Loomis’ (1987) assertion that information is data placed in context, (Schenck & Wilson, 1994). Boland (1987)

introduces an additional thread when he argues that information is a skilled human accomplishment rather than any form of commodity.

This study will accept Tsichritzis and Lochovsky's (1982) definitions while also acknowledging the importance of Schenck and Wilson's (1994) observations. Finally, while recognising the argument for the use of data as a plural, this thesis will accept what has become the more common practice and use it as a singular noun.

## **Models**

The numerous definitions of the word 'model' as a noun provided by the Collins dictionary (Hanks, 1979), are testimony to the usefulness of this word and in many contexts it is clear which meaning is intended. However, at least two distinct interpretations are employed within the context of data representation in IS and considerable confusion can arise when these distinctions are not explicitly discussed. Loosely and Gane (1990) provide a clear, if informal, distinction, suggesting that, "a 'model' can describe two different levels of abstraction...at the first level a model is the thing you build to represent some aspect of your business...at the second level, the set of modelling constructs that you use when creating that model" (p.2).

The first meaning described by Loosely and Gane (1990) is perhaps the most common and the most obvious. It refers to "a simplified representation or description of a system or complex entity" (Hanks, 1979). Here the model is an individual instance of representation, i.e. a specific map, rather than the building blocks that have been used to construct it. These models are constructed primarily as an aid to understanding the underlying slice of reality that they represent which may, or may not, have an actual physical existence (Schenck & Wilson, 1994).

The second way in which 'model' is used is in the sense of a "representative form, style or pattern" (Hanks, 1979). In other words, it is a set of commonly agreed conventions for representing some aspect of the real world. Here the model is an abstraction device that allows the building of artefacts that can be manipulated as if they were real. As Kent (1978, p.93) points out, "A model is a basic system of constructs used in describing reality". The form that the conventions take is less important than general agreement concerning what they are intended to represent. The difficulties in arriving at

this consensus and the confusion that can arise when it is not achieved, or worse still, when it appears, incorrectly, to have been achieved, is an area of interest to this study. The problems inherent in working with an inconsistently defined model will be discussed in various places but particularly in Chapter 5.<sup>1</sup>

Another issue pertinent to this discussion is a concern expressed by Kent (1978) and worth quoting in some detail.

“ [A model] reflects a person’s deepest assumptions regarding the elementary essence of things. It may be called a “world view”. It provides the building blocks, the vocabulary that pervades all of a person’s descriptions” (p.93).

and

“A model is more than a passive medium for recording our view of reality. It shapes that view and limits our perceptions. If a mind is committed to a certain model then it will perform amazing feats of distortion to see things structured that way and it will simply be blind to the things which do not fit that structure” (*ibid.* p.93).

If correct, the notion that the constructs we use limit our perceptions and define the way in which we view the world, is very significant particularly if the models in use are “highly structured, rigid and simplistic” as Kent (*ibid.* p.94) contends. The discussion of these constraints comes from researchers in linguistics and in support of his arguments Kent quotes from Sapir (1931).

“[L]anguage defines experience for us...because of our unconscious projection of its implicit expectations into the field of experience...Categories such as number, gender, case, tense, mode, voice, aspect and a host of others...are not so much discovered in experience as imposed upon it” (*ibid.* p.94).

The possibility that constraints are implicitly imposed by the use of a particular modelling convention is very pertinent to the issues under investigation here<sup>2</sup>. Consequently, further discussion of this idea pervades much of this study and further comments can be found in Chapters 4 and 5.

---

<sup>1</sup> The inconsistency referred to here is when an entity on an E-R diagram can be seen as either the representation of a ‘conceptual ‘ entity or a fully normalised relation, depending either on the purpose for which the model has been constructed or purely on the preference of the modeller.

<sup>2</sup> Once again, it is the notion of an entity. It will be argued that if an entity is viewed as the representation of a normalised relation, a number of constraints are introduced into the conceptual model.

The ubiquitous map analogy used by a number of authors (e.g. Kent, 1978; Loosely and Gane, 1990) is a useful one and demonstrates how both meanings of the word can be combined in general use. Cartographic conventions provide a widely used and useful model (a map), which although it may differ in its representational details, is nevertheless generally understandable. These conventions can be employed to produce a specific map (a model in the first sense), which while it may be idiosyncratic in its use of particular conventions, will none the less be useful, particularly with reference to its specific legend. As an abstraction of reality, much detail is necessarily omitted from such a model and this lack can itself be a barrier to the model's comprehensibility and it is important to be aware of these limitations (Schenck & Wilson, 1994). As Tschritzis and Lochovsky (1982, p.6) remark, "models are used extensively in many disciplines for enhancing understanding and abstracting detail". Some obvious examples are mathematical, statistical and economic models that also take on additional characteristics in that they are "designed to facilitate calculations and predictions" (Hanks, 1979). However as many writers like to point out, the map is not the territory (Korzybski, 1941) no matter how real the model can be made to appear. The tendency to treat models, particularly complex ones, as if they were the real, is neatly highlighted by Kent (1978) in his 'message to mapmakers' although it would perhaps be more appropriate to direct this message to the map-readers. The mapmakers or model makers can be expected to be aware that the "highways are not painted red" (Kent, 1978, p.v). Map-readers on the other hand, particularly inexperienced ones, may not be so clear. This distinction, which reflects significantly on the users' view of models within the development process of an information system, is one that will be revisited later.

### **Data Models and data models**

The previous sections attempt to establish working definitions for the most commonly used meanings of 'model', 'data' and 'information', within the information systems context. Utilising these definitions, it is useful to investigate some terms which use these concepts in combination, particularly, the ubiquitous 'data model'. As with the term 'model' discussed previously, the phrase, 'data model' is used with two distinct meanings; one to describe a **set of conventions** used to represent a simplified and highly abstracted view of data, the second to describe the **specific representation** itself.

Simsion points out (1994, p.22) that when IS practitioners refer to a model, it is almost always in the latter sense while in academic discussions a model will, generally, be understood as the former.

This overloading of meaning is responsible not only for elements of misunderstanding and miscommunication between the two groups but can also make it difficult to discuss both meanings without introducing some form of distinction. Loosely and Gane (1990) suggested a convention of capitalisation to distinguish between the two meanings. They used Data Model (capitalised) to refer to a model of the first type, as in the Relational Model, while, data model (lower case) referred to the second. This convention will be adopted initially here until other possibilities are explored later in this section.

“Data Models are tools<sup>3</sup>” (Kent, 1978, p.194) or “abstraction devices that allows us to see the forest (information content of the data) as opposed to the trees (individual values of data)” (Tschritzis & Lochovsky, 1982, p.5). The concept of a Data Model was initially proposed by Codd (1970) and later defined by Date (1995), as consisting of three components: a collection of object types, a collection of operators, and a collection of general integrity rules. As a tool then, a Data Model is a set of constructs that used together provide the facility to build representations of the data present in an area of interest. The individual items of data represented by the allowed ‘object types’, the relationships between them and constraints on them, represented by the allowed ‘integrity rules’ are fitted together in such a way that they can be manipulated by the allowed operators. Since the introduction of the Relational Model (Codd, 1970) “the data processing community has evolved a number of models in which to express descriptions of reality” (Kent, 1978, p.94). The result being that there are now numerous Data Models (Loosely & Gane, 1990) each purporting to offer significant advantages over its rivals (Brodie *et al.*, 1984).

Brodie *et al.* (1984) provide an early taxonomy of four generations of Data Models: primitive, classical, semantic and special purpose. The first group, consisting of simple file based structures and the last, for use with specialised applications such as CAD/CAM (computer aided design/computer aided manufacturing) will not be

---

<sup>3</sup> Capitalisation added

considered further here. The group of classic Data Models was seen as including “the *hierarchical* (a direct extension of the file model), the *network* (a superset of the hierarchical model) and the *relational*... (a significant departure from both the hierarchical and the network models” (Brodie *et al.*, 1984, p.10). These classifications appear to be largely intuitive and certainly no explanation is given for inclusion of the three models in the classical category. Unlike the formally defined and mathematically based Relational Model, the Hierarchical and Network Models, pre-dating the concept of Data Models as such, were “subsequently defined independently of the languages and systems used to implement them. Before that they were collections of data structures and languages without a defined underlying theory” (Brodie *et al.*, 1984, p.11). The primary shared characteristic of the classical Models would seem to be the existence of commercially available DBMSs (Database Management Systems) based on them. An alternative, and perhaps more descriptive, name for this class might be Implementation Models, i.e. consisting of Models which are commercially implemented as DBMSs.

The third category was semantic Data Models i.e., “Models that are ...designed to provide richer, more expressive concepts with which to capture more meaning than was possible when using classical data models” (Brodie *et al.*, 1984, p.11). With the exception of the E-R Model, a detailed look at the types of Models placed in this category is not appropriate here. However, described in discussion of this class were several features either later incorporated into the Extended-Relational Model or which now form part of the object-oriented paradigm.

To avoid the confusion created by the two uses of ‘data model’ several other terms are used as alternatives to describe a Data Model. Loosely and Gane (1990, p.2) suggest that Data Models are really meta-models but then prefer the term ‘modelling scheme or method’ when referring to a particular meta model. They also use the expression ‘modelling approach or system’ to differentiate between a specific modelling scheme and the generalised characteristics of a type of Data Model. This allows them to refer, for example to “the E-R modelling approach” (*ibid.* p.2). They justify this terminology by the fact that “there are actually lots of flavors of ...E-R and sometimes we do not want to refer to a particular flavor” (*ibid.* p.2). Olle (1993) introduces another possibility, attributed to the ISO Reference Model of Data Management (1993), in the term ‘data modelling facility’ which is “the data structuring rules and data manipulation

rules together.” and which clearly “sets bounds around the concept...to data and the associated processes” (p.46). Yet another alternative is provided by Kim and March (1995), who echoing (Kent, 1978), prefer to use the term “data modelling formalism”(Kim & March, 1995 p.103).

This study will use the last three terms. ‘Modelling approach’ will be used in the sense intended by Loosely and Gane (1990). However, it will generally be used when there is an emphasis on the behaviours and procedures inherent in the development of an instance of a particular Data Model. ‘Modelling facility’ will generally be used in preference to Data Model except when referring to a specific Data Model, e.g. the Relational Model. However, on occasions the terms ‘Implementation Model(s)’ or ‘Paradigm Model(s)’ will be used to refer to Models in Brodie *et al.*’s (1984) classical category, extended to include other commercially implemented Models and their hybrids. ‘Data modelling formalism’ will also be used as a synonym for data modelling approach.

This adoption of an alternative wording for the first meaning of ‘data model’, allows the use of the uncapitalised term ‘data model’ to refer to the individual instances that are created for a specific purpose by the use of a data modelling facility. This is the current practitioner practice and is becoming common in academic papers where these models are also described as ‘application data models’ (Olle, 1993) or ‘user models’ (Loosely & Gane, 1990). There are a number of different kinds of such data models that may be produced within the development life-cycle of an information system and others that are produced as a strategic, rather than a specifically IS, tool. A fuller discussion of these models<sup>4</sup>, what level of detail they may be expected to contain<sup>5</sup>, and which modelling facility<sup>6</sup> may be used to create them is presented in later chapters. However, Martin’s (1990) description of a data model, as “a logical map of data which represents the inherent properties of the data” (p.457), can serve as a general working definition.

---

<sup>4</sup> See Chapter 3

<sup>5</sup> See Chapter 4

<sup>6</sup> See Chapter 9

## Information Models, Schemata and Database Designs

The term 'data model' has been in use, with one meaning or the other, for at least 25 years and remains an essential component of the Information Systems vocabulary. However, perhaps mirroring the transition from data processing to information technology, the term 'information model' is now not uncommon. The origin of the term is not clear (Olle, 1993) although it has been in use since at least 1979 (Kent, 1979). As Olle (1993) points out, it is not unreasonable to assume that "data modelling has something to do with modelling the representations of the data and that information modelling has something to do with modelling the meaning ascribed to the data"(p.47). However, as it appears most frequently as a synonym for 'application data model' (Olle, 1993), it seems more likely to be an early attempt to differentiate between 'Data Models' and 'data models' by renaming the latter.

Where the term 'information model' is explicitly defined as being something other than an application data model, (Flavin, 1981; Schenck & Wilson, 1994; Avison, 1992; Finkelstein, 1989), the distinctions may differ. Avison (1992) uses the expression to describe an organisation-wide view combining "a process-oriented view of the organisation ...with a data-oriented view" (p.62). This is quite clearly intended to describe a model of much broader scope than an application data model and provides a "largely pictorial representation of the organisation in outline" (Avison, 1992, p.62). Flavin's (1981) use of the term is more limited in scope and bears a close resemblance to an application data model, i.e. "a representation of some real-world system that identifies the object types, relationships and operations of that real-world system."(p.116). The difference here is quite clearly the addition of 'operational' information by the inclusion of state transition data, i.e. "pre- and post-conditions attached to the operations" (*ibid.* p.14).

Schenck and Wilson (1994), having identified the importance of the reliable communication of information, emphasise the need for explicit and formal interpretation rules as a component of an information model. "An information model is a formal description of types of ideas, facts and processes which together form a model of a portion of interest of the real world and which provides an **explicit set of interpretation**

rules<sup>7</sup>” (p.10). For them “information modelling is an outgrowth of data modelling” and they agree that defining a distinction between them can be “somewhat fuzzy” (*ibid.* p.10). They themselves identify two areas of distinction. Firstly in the purpose of the activity;

“Data modeling is concerned with specifying the appearance and structure within a computer system of the data which represents particular types of information. Information modeling...has a goal of describing information so that the representative data **could** be computer processed” (*ibid.* p.10).

Schenck and Wilson (1994) continue by making it quite clear that such a model only needs the potential for implementation on a computer, while a data model is explicitly targeted at computer processing. Inherent in these distinctions are the dual roles of a data model as identified by Tschritzis and Lochovsky (1982) which are discussed further in Chapter 4. Secondly, Schenck and Wilson (1994) highlight their treatment of the interpretation rules, which in a data model, unlike their own information model, are, as they assert, “typically implicit; even if they are made explicit, they are informally documented”(p.11).

Finkelstein (1989) on the other hand uses the expression ‘information model’ to describe a data model that captures more meaning than one developed using the classical modelling facilities. He equates a model developed using any of the semantic models e.g. E-R or O-O, as representing an evolution from a data model to an information model. In addition, he asserts that “the process of data analysis (used to develop data models) has also evolved to information analysis (used to develop information models)” (Finkelstein, 1989, p.63).

Veryard (1992) provides a more persuasive case for adopting the term ‘information analysis’ rather than ‘data analysis’ and hence, ‘information model’ rather than ‘data model’. He not only considers the term ‘data’ to be too restrictive but also highlights the use of ‘data analysis’ by other disciplines, such as statistics. As he points out, ‘data analysis’ has traditionally been used “to describe the analysis of data *content*, whereas what we are here concerned with is the analysis of data *form*” (p.24).

---

<sup>7</sup>Emphasis added.

Clearly, there is little agreement on the meaning of 'information model' and the term has been appropriated by various authors to describe methodologies of their own invention. Where it appears to be used synonymously with data model, it appears to carry one of two additional meanings. It is either used to highlight the non-implementation nature of the model and is thus similar to the use of 'conceptual model' discussed in Chapter 4, or it is deliberately used to emphasise the inclusion of operational or behavioural information. No definitive meaning will be attempted here and use of the term will be avoided unless specifically referring to an author's work.

There have been other attempts to differentiate between 'Data Model' and 'data model' by finding alternative terms for the latter, often adding to the general confusion that pervades this area. Many of the early writers, particularly those who use 'data model' in its first meaning, use the term 'schema' to refer to its individual instances (Tsichritzis & Lochovsky, 1982; ISO77, 1977; Brodie *et al.*, 1984; Smith & Smith, 1977a). This, often appearing as 'database schema', was originally used to signify the actual description of the data as it appeared to be stored on the computer and would seem to relate to the dictionary definition of a plan or perhaps, more specifically, a diagram (Hanks, 1979). A wider use of the word emerged with the publication of the ANSI/X3/SPARC draft report in 1975 (ANSI, 1975). The references, in this report, to conceptual, internal and external schemata<sup>8</sup>, extended the meaning away from a plan of the physical data representation and into the realm of 'application' or 'user models', with which it is now largely synonymous. Its most common usage is now in the term 'conceptual schema' where it is often, although not invariably, an alternative<sup>9</sup> for 'conceptual model' (Olle, 1993; Hirschheim *et al.*, 1995). However, it is interesting to note that in most instances where the term 'schema' is used in preference to 'model' there is a tendency to emphasise the data structuring aspects of the representation rather than its communicative role.

The term 'database design' would appear to be self-explanatory and indeed in most contexts requires little explanation. However, it too can serve as a substitute for data

---

<sup>8</sup> See Figure 1 on page 27

<sup>9</sup> A discussion of the differences between the two terms may be found in Chapter 4.

model, although when used in this sense it is almost always qualified in some way. Its most usual use is in the expression 'physical database design' where it quite clearly describes the data structures, as they will appear in a particular DBMS application. In this sense it is synonymous with 'physical schema' and generally represents a combination of the physical data structure diagram and the DBMS commands necessary to create the required database objects. However, it is also occasionally used in the expression 'logical database design' (Kesh, 1995; Shoval, 1985; Storey & Goldstein, 1988) or 'conceptual database design' (Batra & Antony, 1994; Batra & Sein, 1994; Batra & Zanakis, 1994). In these instances it appears to describe a data model which, while implementable, in that it conforms to a Data Model, has not been tuned for implementation with a specific DBMS, e.g. a model expressed in general relational terms but not tailored for a specific database implementation, such as DB2<sup>10</sup>. To complete the circle of confusion this has been called by some authors, often those writing about the process of creating data models, a 'logical model' where the meaning is similar to, but not synonymous, with the 'conceptual schema' of the ANSI/X3/SPARC architecture discussed in the next chapter.

Of the terms discussed above, 'conceptual schema', 'conceptual model', 'logical database design' and 'logical model' will all be used in this study. While the intended meaning will generally follow those given above, further explanations and distinctions will be made during the discussion of the data modelling architecture in the following two chapters.

### **Subject area, problem domain and universe of discourse**

Although not generally leading to confusion, a number of different terms have also emerged to describe the area under consideration by the modelling process. A report by an ISO committee introduced the expression, borrowed from Wittgenstein (Olle, 1993), 'universe of discourse' to describe it (van Griethuyzen, 1983) and this, with a usual abbreviation to UoD, has become the standard terminology within NIAM. This is referred to as the "object system" by Hirschheim *et al.*, (1995 p.15). Olle (1993) prefers the "more down to earth term "subject area" (p.47) while Flavin (1981), amongst others,

---

<sup>10</sup> DB2 is a registered trademark of the IBM Corporation.

uses 'problem domain'. There appears to be no substantive differences between the terms and this study, while favouring UoD will use them interchangeably.

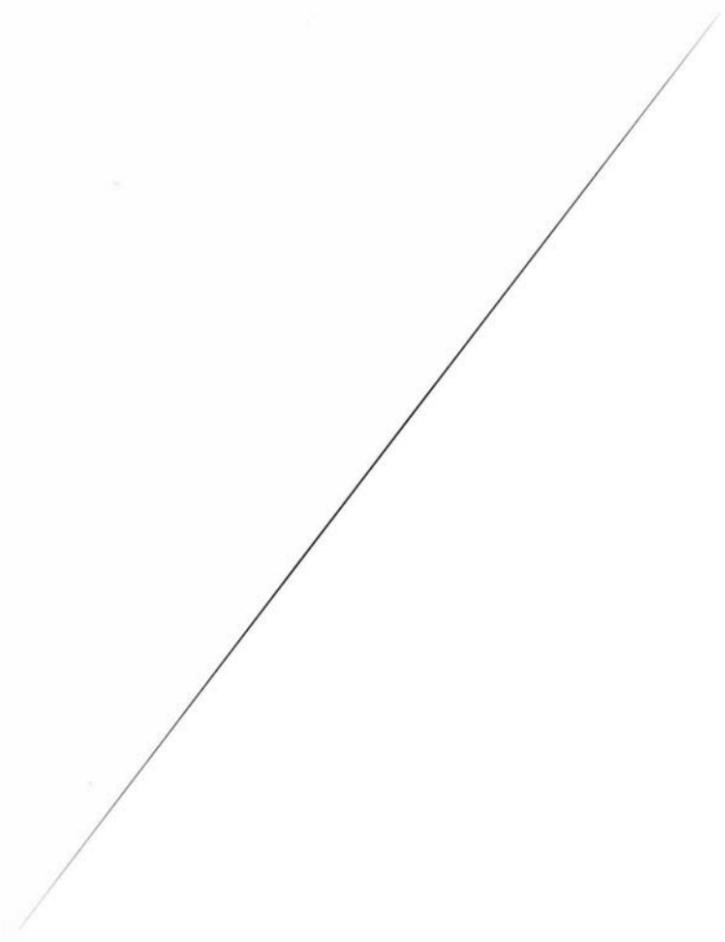
### **Entities, Objects, Constructs and Things**

In this study the term 'construct' will be used to refer to the building blocks provided by a data modelling facility or Data Model, i.e. the elements from which a data model may be constructed. Most of the individual constructs provided by the two modelling facilities under consideration will be identified and discussed, as they become relevant, in the main part of the study. Two however, entities and objects, require some initial clarification.

In general, 'entity' and 'object' will be used interchangeably in this study in the sense of "something having real or distinct existence" (Hanks, 1979), in preference to the vague and rather unsatisfactory 'thing'. At times however, they will be used in a more specific sense. An entity will sometimes refer to the specific construct provided in the E-R Model, i.e. as an aggregation of properties. Similarly an object may sometimes refer to the equivalent construct in either the NIAM-CSDP or the Object-Oriented approach. The context will generally make it clear which meaning is intended.

This leaves two terms that require further discussion. 'Data modelling', referring to the entire process of creating data models for a specific domain, and 'conceptual data modelling', a subset of this activity, are treated separately, and in some depth, in the following two chapters.

4



# 3 Data Modelling

*"I am frequently asked by project leaders and managers,... "What are the benefits of data modelling?" The simple answer is that data modelling is not optional; no database was ever built without at least an implicit model... The choice is not whether or not to model, but whether to do it formally" (Simsion, 1994 p.21).*

## A Three Level Data Architecture

In 1975 the Standards Planning and Requirements Committee of the American National Standards Institute Committee on Computers and Information Processing published an interim report (ANSI, 1975) containing proposals that have shaped and guided the development of database applications for the last twenty years. Most significant was the observation that the data within a particular subject area could be usefully viewed as having its own inherent structure. This structure was not necessarily identical to the physical structure of the same data, as it was stored on a computer, nor the same as the conceptual structure of the data as seen by any one individual user. This insight provided the basis for the development of a three level architecture, which viewed data at three different levels of abstraction (Figure 1). It also enshrined a guiding principle of subsequent database development; that users of the database should not need to be concerned with any level of abstraction of the data other than their own external schema.

The three levels of abstraction identified in the American National Standard Institute (1975) report also highlighted the possibility and desirability of achieving a level of data independence far greater than had been previously envisaged. The external schema, which represented the individual user's view of the data, was not mapped directly to the internal schema, the physical database structures in which the data was stored. Instead both were mapped to an intermediate level, namely the conceptual schema. This, created by the database designer, provided a global or community view of all the data in the information system (Date, 1995) and was actually the superset of all the external

user views. The benefits of this arrangement were numerous (Ricardo, 1990) and hinged on the idea that the conceptual schema acting as a buffer, could hide modifications in the other two levels from each other. Protecting the external schema from any changes to the physical data structures provided physical data independence. Provided that the mapping between the internal and conceptual schema could be maintained such alterations did not need to impact on the users' view of the database.

This principle also worked at the level of logical data independence. Here, changes in the requirements of an individual user did not necessarily result in changes elsewhere. Even where a new requirement entailed the need for additional data to be added to the internal schema other users need not be impacted by the change. The adoption of this architecture, particularly where much of the mapping could be maintained by the DBMS itself, was crucial to the development of the database approach to information systems and continues to play a fundamental role today (Avison, 1992).

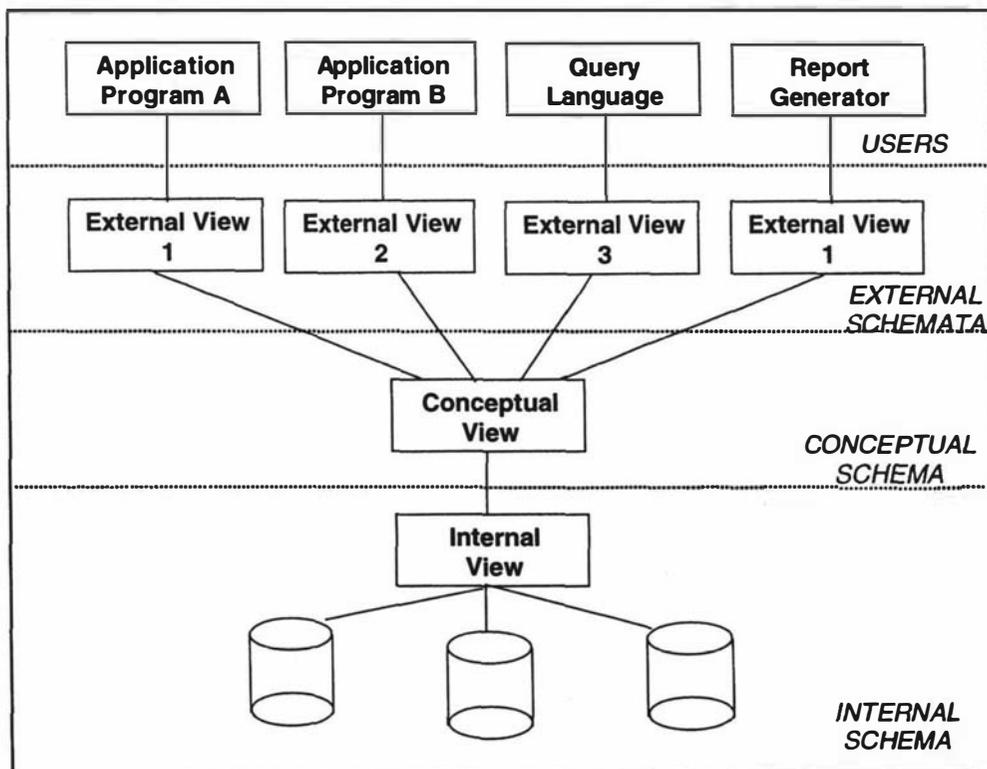


Figure 1. ANSI/X3/SPARC Architecture - adapted from Avison (1992)

## A Database Design Process

The three level architecture, however, not only provided a new framework in which to create and view data representations but also provided implicitly, a template for a four-stage database design process. Within this process the database designer should: -

1. seek to ascertain each user's view of the data, i.e. **analyse** each user's data **requirements**,
2. amalgamate these views, i.e. **design the conceptual schema**,
3. create physical structures in which to store the relevant data, i.e. **create the physical database design or internal schema**, and
4. reproduce the original views for each individual user; i.e. **create the external schema**.

By implication, all the users were required to do was describe the information requirements of their view. Eventually they would be provided with an interface to the data that accurately matched their initial specification. This process, including the distinction between data analysis (step 1) and database design (steps 2 - 4), was formalised by Teorey and Fry (1982) and has remained the general approach to database development ever since. It can be found in many recent textbooks (e.g. Avison, 1992; Kroenke, 1992; McFadden & Hoffer, 1994; Ricardo, 1990; Sanders, 1995; Simson, 1994) and systems development methodologies (Eva, 1994; Finkelstein, 1989; Page Jones, 1988).

This rather optimistic view of the users' role in the process was perhaps never a reality and has certainly proved an elusive goal for database designers, although some exponents of NIAM would argue that their approach comes very close to this ideal (Sharp, 1994). However, in the pre-relational environment in which the ANSI/X3/SPARC (1975) report was produced, the 'users' were largely, either computer programs or computer programmers. In neither case was it necessary to undertake extensive validation of whether the representation of the data structure that the designer had created, matched the users' own view of the data structure. The significant communication was almost entirely one way, from the users to the designer. The majority of the communication in the other direction was expressed within the success or failure either of the programs to run, or of the programmers to write the programs to the specifications they had been given. In a case where the data structures did not successfully support these 'user' requirements, it may often have been easier to adjust

the requirements rather than to change the database structure. However, with the advent of relational database management systems in the early 1980's, and the later addition of personal computers, the situation changed quite dramatically. The programs and programmers were increasingly overtaken in numbers by real 'users' of the data, i.e. people with relatively few technical skills but extensive enterprise knowledge who now had both the opportunity and the desire to directly access the data of interest to them.

This alteration in the nature of the users also signified the need for a major change in the database design process. For the first time database designers were required not only to understand and record the data requirements of users for the benefit of other IS practitioners, but also to explain to human users the data representations that they had created. This aspect of the database design task became increasingly important. It became necessary both to ensure that the sometimes conflicting requirements of a number of users had been correctly understood and combined, and also to provide the users with a documented structure that could inform their own navigation and use of the eventual database. It was no longer sufficient for designers to express their decisions in terms that required significant technical expertise to understand they now had to produce documentation and explanations that were comprehensible to a wider non-technical audience. In terms of the ANSI framework, some users needed to understand enough of the conceptual schema, from which the internal schema would be constructed, to be able to validate the structures it was representing.

This process of describing and analysing data, and designing the requisite data structures is referred to in general terms as data modelling<sup>1</sup>, while the resultant outputs of the process are known as data models. As Kent (1978) observed all data models act as bridges in the sense that they are "techniques for representing information and are at the same time sufficiently structured and simplistic to fit well into computer technology"(p.93). However, as Kent also recognised, "Most models describe data processing activities not human enterprises. They pretend to describe entity types but

---

<sup>1</sup> The same process may also be termed 'database design'. Although these terms are often treated as synonyms, the specific choice made by an author often provides an insight into the perceived *intention* of the process.

the vocabulary is from data processing” (Kent, 1978, p.96). This tendency may also be viewed in the light of the development of the practice of data modelling.

### **Instantiating the Architecture**

At the time of the American National Standard Institute (1975) proposals, only limited forms of recording the various levels of the framework existed. The external schemata were represented by the paper forms, screens, reports or program file structures that the system used or produced, while the conceptual schema had no tool other than natural language to support it. Only the internal schema had any means of formal description. These diagrammatic representations of network and hierarchical database structures<sup>2</sup> (Bachman, 1969), were the first ‘data models’ to see widespread use. By providing a visual representation of complex physical data structures they assisted database designers and database administrators to create mental maps of the data structures that they were managing and manipulating. Developed and designed for use by people already familiar with the concepts and constraints of the Data Models that they were reflecting, Bachman diagrams were neither intended for, nor suitable as, communication tools between designers and non-technical users. In Kent’s (1978) terms, they did provide a bridge but only between the highly technical understanding of the designer and the complex physical structure that was being created.

As Hirschheim *et al.* (1995) observe, although it was never entirely clear what the ANSI/SPARC report meant by the ‘unified’ conceptual level and by the conceptual schema, the report created a great deal of momentum to the development of Data Models that were by nature ‘conceptual’ or ‘semantic’ (e.g. Hull & King, 1987). The most influential of these, the E-R Model, defined by Peter Chen in 1976, appeared to provide a relatively simple and intuitive means to fill a gap in the ANSI framework and, in so doing, revolutionised the database design process. The E-R Model provided a representation of data which did not reflect the constraints of the physical (hierarchical and network) database structures and which as a number of studies (Konsynski, 1979; Brodie *et al.*, 1984; Yao *et al.*, 1984; Teorey *et al.*, 1986; Batra *et al.*, 1990) later

---

<sup>2</sup> Bachman’s work also led to the formal definition of the network data model by the CODASYL Database Task Group in 1971 (CODASYL, 1971).

demonstrated, appeared to provide a clear and intuitive way of communicating with non-technical users. However, not only did Chen's (1976) paper describe a useful abstraction tool it also provided a straightforward diagrammatic means of representing the structures it was describing. Within a short time, E-R diagrams became the most commonly used means of representing the conceptual schema, and Date (1995) goes so far as to suggest that the E-R Model's lasting popularity is probably due more to the existence of this diagramming method than to any other factor. However, the construction of both Bachman and E-R diagrams remained a technical (i.e. data processing) task and a considerable degree of skill was required to map the E-R diagrams to the physical structures.

With the advent of commercially available relational database management systems (RDBMS), E-R modelling appeared to have consolidated and justified its pre-eminence. The E-R Model utilised some constructs that mapped naturally to relational objects, i.e. entities to relational tables and attributes to columns. While other constructs, for example relationships required more complex translations, in general, the process of mapping between the conceptual (E-R) schema and the internal (relational) schema was significantly more straightforward than before. No diagrammatic method was provided for representing the Relational Model, which as some studies have suggested, reduced both its accessibility and comprehensibility (Amer, 1993; Batra *et al.*, 1990; Batra & Srinivasan, 1992; Jarvenpaa & Machesky, 1989; Jih *et al.*, 1989; Juhn & Naumann, 1985). Smith and Smith (1977b) found it necessary to create their own diagrammatic notation but gradually a subset of the E-R notation was used to produce representations of relational schemata or 'logical database designs' as they were termed.

Alongside these developments, relational technology also created a significant shift in the perception of the internal schema. Database designers and administrators were buffered from many of the actual physical structures by the DBMS itself. In many situations, technical staff could, like the users, view the database as a collection of tables rather than in terms of files, machine addresses and the like. As a result, both technical and non-technical users came to utilise some form of the E-R diagrammatic notation as a useful means of recording and understanding the physical database as well. These pragmatic adaptations while effective in improving communication about relational data structures nevertheless had some important implications.

## **The development of the E-R/Relational Hybrid**

The apparent similarities between the technical and non-technical views, the use of E-R diagramming conventions for both conceptual and internal representations of the data and the tendency for data modelling to remain a task undertaken by technical staff, may have been largely responsible<sup>3</sup> for the development and widespread use of what can be termed an E-R/Relational (E-R/R) hybrid (Atkins, 1996). This hybrid, albeit unacknowledged, is promoted by writers, such as Simsion (1994) and Benyon (1997) and utilised by a number of CASE tools. It uses E-R, or more commonly, extended E-R constructs (Cattell, 1994) and notation to provide a graphical representation of a normalised relational structure, is generally characterised by being in third normal form, having no relationship attributes and resolving all many-to-many relationships. Such a model is often termed an E-R model, or sometimes an E-R diagram, but clearly does not conform to Chen's (1976) original definition and it is this hybrid that should perhaps be the target of many of the criticisms currently levelled at the E-R Model. The largely unrecognised development of this hybrid certainly appears to have contributed to the general confusion surrounding the purposes and format of the conceptual schema.

## **A Revised Architecture**

As Shoval (1985) recognised, these developments necessitate an updating of the original ANSI 3-level architecture to reflect the following changes: -

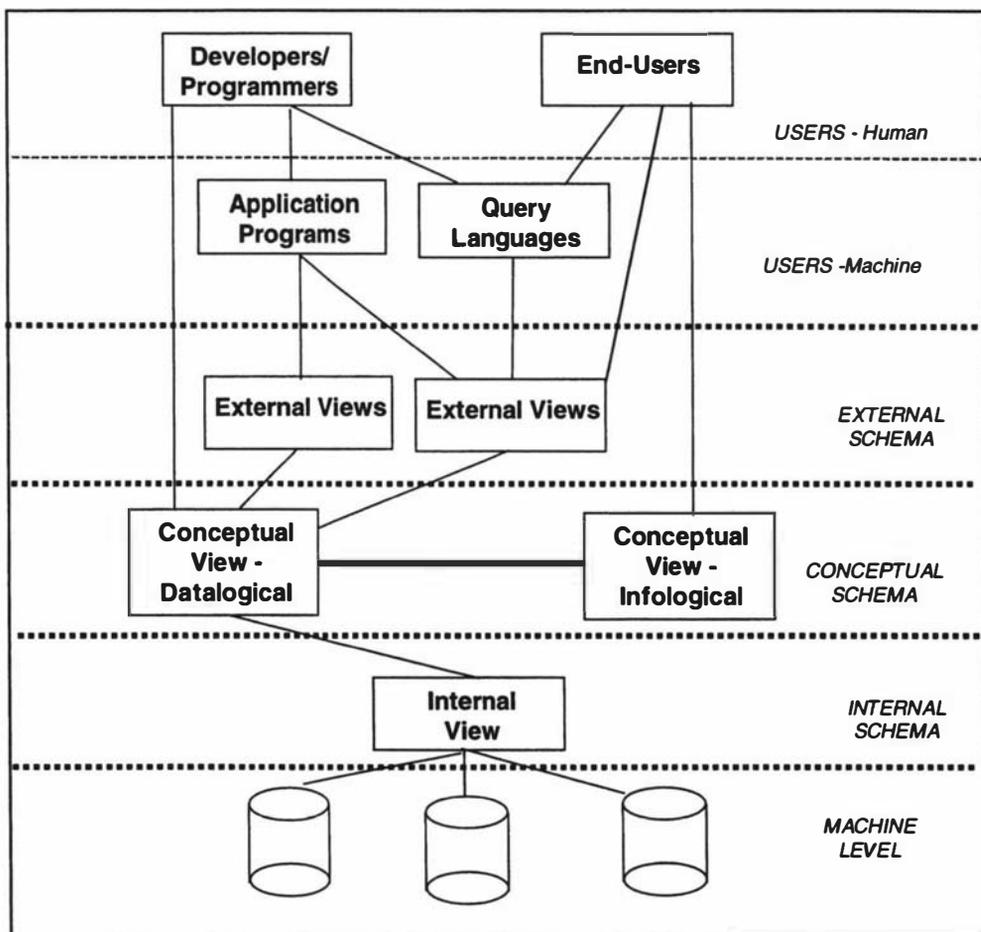
- a greater distance now exists between the internal schema, as viewed by the database administrators, and the machine level view,
- the conceptual schema plays two discrete roles, one as originally envisaged by American National Standard Institute (1975), the other required by the disparate and non-technical nature of the new users who must interact with the database (van Griethuyzen, 1982), and
- the recognition that these two roles need not necessarily be fulfilled by the same representation (Tsichritzis & Lochovsky, 1982)

---

<sup>3</sup> The use of CASE tools may also have played a part but is outside the scope of this study.

A redevelopment of the ANSI framework incorporating these changes is presented in Figure 2. It differs from the '5-schema framework' outlined by Shoval (1985), which seeks to integrate the ANSI framework with the recognised stages of database design, by having "the original ANSI/SPARC 3-schemata at the DBMS implementation level, and on top of them ...the...Conceptual Schema (at the semantic or analysis level)...and the database schema (at the logical or design level)" (p.417).

The framework illustrated here is intended to address a number of issues of which the most significant is the separation of the conceptual schema into two views each intended to serve a different purpose. These views have been termed the 'infological' and the 'datalogical', following terminology first used by Langefors (1963) and, later, applied to the functions of data models, by Tschritzis and Lochovsky (1982).



**Figure 2. A revised view of the ANSI 3 level architecture**

The 'datalogical' view refers to the way in which the data, required by an information system, is to be structured for efficient electronic processing. Thus, the **internal view** is

by definition, datalogical, in that it represents the actual or potential physical schema in all its detail. For example, in a typical relational system, the internal view will contain information such as the specific names and attributes of the files in which the tables, indexes and integrity constraints reside together with the names, addresses and properties of system files. In general, this information will only be used by the systems and database administrators.

The **datalogical conceptual view** is also a representation of the physical structuring of the data but at a higher level of abstraction. Where the physical database is existent, the datalogical conceptual view may contain a number of constructs of use to system developers and programmers, such as tablespaces and indexes but, in general, this view will primarily reflect the tables in which the database users perceive the data to be held. Both human and machine users (such as programs and query languages) will be able to utilise this view of the database for almost all their needs and directly manipulate it as if the data really was stored in these structures. However, in many cases a subset of the complete datalogical view will be provided in the form of an **external view**, both to protect the underlying data from breaches of security and also to reduce the complexity of the 'problem space' that users need to comprehend.

While the datalogical conceptual view is concerned with the representation of the structures by which a system's information needs may be met, the **infological conceptual view** is concerned with the content or the semantics of those information needs (Hirschheim *et al.*, 1995). Its primary function is to provide a representation of the target information needs in a form that is as unconstrained as possible by physical considerations and is easily accessible by non-technical users. It is essential that there is a direct mapping between the two views (emphasised by the bold line that links them in the diagram) that together make up the conceptual schema. Although the infological view is less important once a physical database has been constructed, during the development process, and any subsequent modification, it provides a representation scheme whereby users can verify the validity of the information needs that are being represented in the database without having to understand the structures by which those needs are being met.

## A Meta-Data Architecture

Data modelling, then, may be defined as the process of creating representations of data within a specific problem domain in a formalised and organised way. It is an activity considered by many (Hitchman, 1995; Sanders, 1995; Batra & Davies, 1992; Simsion, 1994, among others) as fundamental to the creation of any information system, no matter of what size or what type. However, for many organisations the process has developed and broadened significantly since 1975. The number, the complexity, the purposes and the perspectives of the models that are produced can vary considerably. The scope of the modelling activity may also differ, ranging from a specific application database schema, to a model of the entire organisation, a 'corporate' or 'enterprise model', which will never be implemented electronically. Consequently, the data modelling activity may be restricted to produce only a physical database design, or it may use a variety of tools and techniques to produce a number of models, each serving a different function. The original ANSI framework and the extensions to it explored earlier, are concerned with characterising the architecture of a single data model and as such are not able to describe the relationships between the various models that may now be produced by a sophisticated data modelling exercise. Instead a framework for categorising these models, termed here a meta-model architecture, is required. This is partly to facilitate discussion about the purposes of the different types of models and also to assist in clarifying the meaning of 'conceptual model' as it widely used within the practitioner community and as it is used within this study.

There is little recognition of this need in the academic literature. McFadden and Hoffer (1994) describe a similar architecture to that illustrated at Figure 3, although their equivalent paradigm model is DBMS specific while their physical design is concerned with "storing record formats, selecting access methods and deciding on physical factors such as record blocking"(p.241). However, there are references to such an architecture in the practitioner literature (Wong, 1995; Zachman, 1987) and it has also been observed by the researcher in the course of her work with various organisations.<sup>4</sup> This proposed meta-model framework is intended to highlight the importance, the

---

<sup>4</sup> Particularly, Statistics Division, Inland Revenue UK. 1985 - 1992

perspective and the changed nature of the conceptual model and its potential position in the data architecture of the post-relational era.

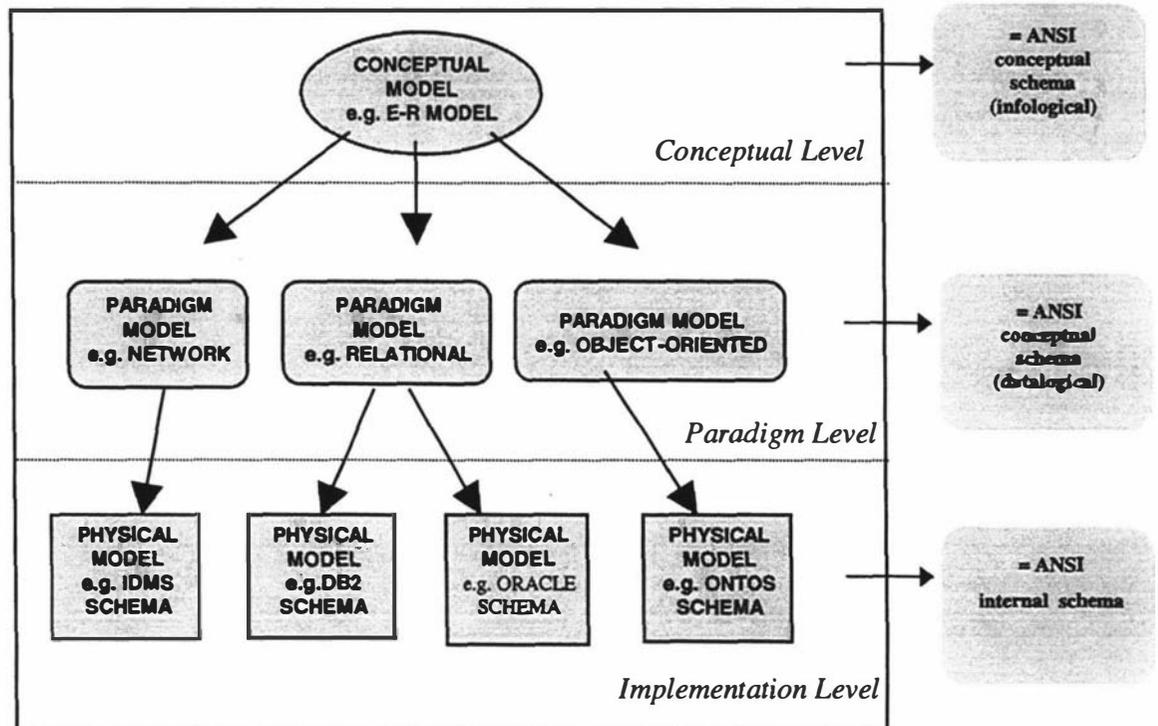


Figure 3. Meta-Model Architecture

The meta-model architecture includes a **conceptual model**, semantically close to the subject area, and partly serving the infological function of the conceptual view of the revised ANSI schema, presented in Figure 2. This model is intended to provide a high-level, implementation independent view of the data, which can be understood by both technical and non-technical users alike. While it is necessary to structure the data in order to provide a useful view of it, there is a general consensus that this view should be as unconstrained as possible by implementation considerations. The emphasis would appear to be on capturing and representing a comprehensive but comprehensible view of the data, rather than moulding it to any specific form of electronic storage. It will be argued later, in Chapters 3 and 4, that the primary activity in the construction of a model at this level is thus one of analysis. It has been suggested, that some form of Semantic Model (e.g. Peckham & Maryanski, 1988) is the most appropriate tool and much academic literature (e.g. Shoval, 1997) and many textbooks (e.g. Avison & Fitzgerald, 1995; Burch, 1992; Ricardo, 1990; Connolly *et al.*, 1995; Mannila & Rähkä, 1992)

presume that the E-R Model is the most widely used. However, if most E-R models are actually E-R/R hybrids then they are clearly not independent of implementation considerations and users will require some understanding of relational structuring rules, particularly normalisation, in order to provide semantic verification.

From this conceptual model any number of different implementation or **paradigm models** can then be derived. These models are broadly equivalent to the pre-construction, datalogical conceptual view illustrated in Figure 2. The purpose of this model is to design data structures that will accommodate the conceptual model in a form that is generic to a Data Model. Thus if the target database is relational, the paradigm model will be a relational model. If the conceptual model is implementation independent it should be possible to transform it into any paradigmatic structure. There may, indeed, be a need to transform part of the conceptual model to one paradigm and part to another. The primary emphasis of this model is thus on design.

The relevant **physical model**, designed from the appropriate paradigm model, will reflect the changes to the 'pure' paradigm model necessitated by the constraints of the actual DBMS package that is used for implementation. It will include a description of all the objects, such as indexes, recovery logs and triggers, resident in the physical database. It thus covers both the internal schema and the machine level of the revised ANSI architecture. However, if it is a relational database at least, it will, post-construction, also provide the datalogical conceptual view of the tables (or virtual tables) with which the users can interact.

Aside from the benefits that can arise from having a clearly delineated purpose for each level of the meta-model architecture, it also provides a form of independence similar in effect to that provided by the use of the ANSI 3-level schema. At the organisational level, the use of such a meta-model architecture provides for several implementations of the same data area in different paradigms or different versions of the same paradigm, a useful consideration in terms of the data integration that may be required following company take-overs or mergers. While the models at the lower levels may be very different, they will be reconciled at a higher level, at the paradigmatic level for implementation models of the same paradigm or at the conceptual level for models of different paradigms. Quite aside from any technical benefits, many larger companies

have come to recognise that all data, whether stored electronically or not, is a valuable organisational resource (Kroenke, 1992). In order to both protect and manage this resource effectively, it is necessary to have a clearly documented record of what data the organisation owns. As a result 'corporate' or 'enterprise' models (CCTA, 1994) are sometimes created in an attempt to capture this information at the conceptual level. In some cases, this view is integrated with the paradigm and/or implementation levels, while in others it remains an isolated structure. There are also benefits to be gained at the system level. Here the existence of models at all three levels of the architecture can ease the process and reduce the cost of migrating either from one paradigm to another, e.g. from Relational to Object-Oriented or from one DBMS package to another, e.g. from DB2 to Oracle.

Once again this meta-model structure partially reflects the historical development of data modelling. Initially the need for understanding the data structure was engendered by the need to create an effective database and this would result in the creation of a physical database design (*implementation level*). This was followed by the requirement to build a DBMS-independent view of the data structure in order either, that a number of databases using different DBMS (of the same paradigm) could be created or integrated, or to provide a less technical data view for verification by users (*paradigm level*). Currently many organisations have a requirement to build some form of data framework either to aid in the construction or integration of databases of the same data but different types (e.g. Object-Oriented and Relational) or as described above as a mechanism for managing the data resource (*conceptual level*). However, it is possible that the use of the ubiquitous E-R/Relational hybrid to represent data at any or all of these levels, either separately or in combination, has added greatly to the confusion surrounding the nature and purpose of the different levels themselves.

## **Summary**

It is evident that some of the objectives of data modelling have changed since the ANSI architecture was first proposed. The design of structures suitable for storing data relevant to a specific application or system, while still the primary purpose, has been transcended by the need to understand, document and communicate about the data resource in a broader arena. This secondary, infological purpose while recognised (e.g.

Campbell, 1992; Kent, 1978; Tsichritzis & Lochovsky, 1982) has too often been under-emphasised both by academics and practitioners alike. Alongside these developments there has also been a growing trend to use “data analysis...much earlier, in the planning and requirements definition stages, as one of the techniques for defining the problem domain itself”(Darke & Shanks, 1995b p.4-5).

Another interesting and relevant issue, which also serves to highlight the confusion that seems to surround data modelling, is the nature of the activity itself. The discussion here has hinted that the creation of the infological view of data is primarily an analysis activity. However, Simsion (1994) argues forcefully that data modelling is primarily a design activity and, therefore, inherently creative. He argues that it is misnamed under the designation ‘data analysis’. Many authors, particularly of textbooks, describe the data modelling process as consisting of two distinct parts, data analysis and database design, while they describe the same tools and techniques as being appropriate to both. The nature of the modelling process and the concomitant issue of creativity are discussed in the following chapter.

There are a number of other issues directly related to data modelling that have not been emphasised<sup>5</sup>, if mentioned at all, in the preceding discussion. Although some of these will be discussed at appropriate points others, such as the effect of CASE technology on the modelling process, while recognised as important, are beyond the scope of the present study.

---

<sup>5</sup> e.g. the practical difficulties inherent in forming a consensus view and in creating universal definitions for context-dependant entities.

# 4 Conceptual Data Modelling: some underlying issues

*"Suddenly, you discover that the map is not the territory, the menu is not the meal. The word is not the thing. Words are only symbols. Concepts are models of reality built out of words. We discover that we do not live in reality at all. We live only in a well constructed model of reality - a model that we've been constructing since birth - a reality built out of words. We live in language...and our language shapes and colors our experience" (Gerrold, 1988).*

## Introduction

While the importance of constructing a conceptual data model is generally accepted (e.g. Batini *et al.*, 1992; Laender & Flynn, 1994; Pavlia *et al.*, 1992), there are differing opinions concerning its nature and purpose. Previous chapters have attempted to lay the groundwork for a more detailed investigation of some of these differences.

Author	Date	Data	Reality	Objective	Subjective	Infological	Datalogical
Shave	1981		✓	✓		✓	
Elmasri <i>et al.</i>	1985		✓			✓	✓
Shoval	1985		✓				✓
Storey & Goldstein	1988		✓				✓
Bock & Ryan	1993	✓					
Jarvenpaa & Machesky	1989		✓				
Creasy & Moulin	1992		✓	✓			
Amer	1993	✓		✓			✓
Tjoa & Berger	1993		✓				✓
McFadden & Hoffer	1994	✓					✓
Shoval & Frumermann	1994		✓				✓
Moody & Shanks	1994		✓		✓		
Kim & March	1995		✓		✓	✓	
Calway & Sykes	1995		✓				
Siau <i>et al.</i>	1995		✓			✓	✓
Avison & Fitzgerald	1995	✓			✓	✓	

**Table 1. Some definitions of conceptual data modelling.**

From the confused and often contradictory discussions in the literature, which often display a lack of rigorous definition of the term 'conceptual data model', three major themes, which are the subject of this chapter, seem to emerge. Two of those themes, 'reality vs. data' i.e. what is being modelled and 'objectivity vs. subjectivity' i.e. what

can be achieved by the modelling process, concern the nature of the representations. The third issue, *'infological vs. datalogical'*, stems from the purpose or intention behind the representation's construction. Many of the definitions provided or implied by researchers in the area can be classified within these three themes and Table 1 provides a summary. However, not all the definitions are clear or consistent and, consequently, the table represents a necessarily crude and subjective analysis.

### **Reality vs. Data**

The ANSI architecture clearly identifies the conceptual schema as a representation of the stored data, free of the technical details of physical implementation (ANSI, 1975). In this sense, the conceptual schema comes closest to the more generalised definition of "an organiser...that shows how the parts and operations of a system fit together" (Mayer, 1989 p.61). However, the database design process, described earlier, which grew from this architecture, together with the ready adoption of the E-R model as a useful tool for building it, cast the conceptual schema into a somewhat different role. Instead, it became a means of formalising the results of the 'requirements discovery' phase (McFadden & Hoffer, 1994). Thus, gradually, the conceptual schema came to be seen as a representation not only of the data structures of the database but of the superset of the users' views, of their 'reality'. This subtle shift in perspective had significant implications for the ways in which conceptual schemas were both built and used. As early as 1978, Kent was warning,

"One thing we ought to have clear in our minds at the outset of a modelling endeavour is whether we are intent on describing a portion of reality (i.e. some human enterprise) or a data processing activity. Naming rules do not reflect the conventions we use for naming people and things, they reflect instead techniques for locating records in files. Failure to make the distinction leads to confusion regarding the roles of symbols in the representation of entities and some mixed ideas of domain." (p.96)

Nevertheless, the perception of the conceptual schema, in this new incarnation more commonly called 'the conceptual model', as a formalised representation of the users' 'enterprise reality' (Kim & March, 1995) or the "user's views of the world" (Jarvenpaa & Machesky, 1989 p.367), persisted and grew. The increased need for database designers to communicate with non-technical users no doubt fostered this change. Consequently, it is now unusual to find the conceptual model described as "a representation of the entities, data items and the associations between entities stored in a

database" (Amer, 1993 p.2), or as the synthesis of "various user views...into a global data base design" (McFadden & Hoffer, 1994 p.241). Most recent researchers have a view more consistent with the description of "formalizing and representing the data structures of reality" (Shoval & Frumermann, 1994 p.28) or "the conceptual representation of the enterprise" (Siau *et al.*, 1995 p.341).

This latter group generally considers the E-R Model to be an appropriate tool for conceptual data modelling, echoing Chen's (1976) belief that "the entity-relationship model adopts the more natural view that the real world consists of entities and relationships"(p.9). Indeed, much of the current justification for the use of the Object-Oriented Model also emphasises this apparent advantage. Martin and Odell (1992), for example, summarise many of these arguments when they say,

" The models we build in OO analysis reflect reality more naturally than the models in traditional systems analysis. Reality, after all, consists of objects and events that change the state of those objects. Using OO techniques, we build software that more closely models the real world" (p.67).

The distinction between these views of what is being built is clearly important both in terms of modelling behaviour and in terms of model perception. If the model is designed to reflect the form of data storage structures, whether actual or potential, there is only one possible version, the map of the database itself. Modelling behaviour will reflect this certainty, as creating the model will involve either a direct transformation of the existing database structures into a higher level of abstraction (a form of reverse-engineering) or will entail selecting potentially useful database structures from the users' world. It is not surprising that those who subscribe to this view appear comfortable with the use of the Relational Model as an appropriate means of recording a conceptual model (Amer, 1993; Avison, 1992; Howe, 1983; McFadden & Hoffer, 1994). Indeed, Avison (1992) is able to entitle a chapter 'The Conceptual Schema: Data Analysis and the Relational Model' with no acknowledgement of any possible contradictions in his choice of words. However, if the model is viewed as a map of reality, with no consideration of 'how data is stored on computerised files' (Jarvenpaa & Machesky, 1989; Batra & Antony 1994; Bock & Ryan, 1993; McFadden & Hoffer, 1994; Navathe, 1992; Schenck & Wilson, 1994; Shave, 1981; Shoval & Frumermann, 1994; Siau *et al.*, 1995; Storey & Goldstein, 1993), then the modeller's own personal, philosophical view of reality is a significant factor (Klein & Hirschheim, 1987).

However, a number of authors avoid committing themselves clearly to either position, preferring the more ambivalent term ‘information requirements’ (e.g. Batini *et al.*, 1992; Laender & Flynn, 1994; Shanks, 1997; Storey & Goldstein, 1993). This term appears to occupy the middle ground by acknowledging that what is being modelled is neither a ‘true’ portion of reality nor a direct copy of a database structure. Instead it is a means of capturing a collection of data elements required by a user to fulfil certain functions, and representing them in a manner that will facilitate their electronic storage. This middle position also recognises that “although the data may have other meanings which are hidden, unknown or even irrelevant, the meaning captured by the data model should be adequate for the purpose required” (Tsichritzis & Lochovsky, 1982 p.6).

Avison and Fitzgerald (1995) succinctly summarise this middle ground and thus, not only provide the viewpoint that is adopted in this study but also comment on the issue discussed here in the following section. They write,

“The data model can only be *a* model and not *the* model of that part of the real world being investigated. It cannot reflect reality completely and accurately for all purposes. Even if data analysis has ‘gone according to plan’, the resultant data model cannot objectively represent the organisation. It is a subjective view distorted by the perceptive process...however, the data model...usually proves in practice to be suitable for the purpose of building a database” (*ibid.* pp.69-70)

### **Objective vs. Subjective**

Inherent in the various views described in the previous section, are the data modelling paradigms alluded to by Kent (1978) and clearly identified by Klein and Hirschheim<sup>1</sup> (1987). These paradigms, broadly falling into either the objectivist or the subjectivist position, have “implications for the interpretation of the UoD and consequently for the interpretation which one gives to data models” (Klein & Hirschheim, 1987 p.9). In broad terms, the objectivist view holds that entities exist in the real world and that a modeller’s task is to uncover those relevant to the UoD. On the other hand, the subjectivist position holds that each person has their own individual view of reality and that the appropriate activity is to recognise multiple realities and adopt an interpretivist stance (de Carteret &

---

<sup>1</sup> Burrell and Morgan (1979) developed the original framework. Klein and Hirschheim applied it to data modelling.

Vidgen, 1995). As Klein and Hirschheim (1987 p.9) observe, “the difference is whether one believes that a data model ‘reflects’ reality or consists of subjective meanings and thereby constructs reality”. Shoval and Frumermann (1994) are clearly objectivist in stating that “the conceptual schema...portrays the data structure of the reality being modelled”(p.28). Likewise Keuffel’s (1996) comment that, “ the process of creating a model is an attempt to capture the essence of things both concrete and abstract, to make order of the chaos inherent in the world around us”(p.83). MacEachren (1995) on the other hand clearly reflects the subjectivist position, when he states, “when we build these abstract representations...we are not *revealing* knowledge as much as we are *creating* it” (p.v). Tolis (1996) summarises these positions in terms of “the relation between the model and the thing modelled” as follows,

- “An *objective* assumption is characterised by a focus on structures in the world. Models are viewed as expression of facts, not capable of influencing the thing modelled. They are valued in terms of correspondence.
- A *subjective* assumption is characterised by a focus on structures in the human mind. Models are viewed as expressions of values, capable of influencing the thing modelled. They are valued in terms of beauty (simplicity, elegance)” (p.741)

Moody (1998) and Moody and Shanks (1994) view data modelling not as a “deterministic process of uncovering the ‘correct data model’, but a process of searching for alternative solutions”(p.2) and this comes close to encapsulating the behavioural differences implicit in these distinct positions<sup>2</sup>. The objectivist belief in one ‘correct’ data model leads modellers to perceive any problems with a model as “a failure to capture and specify the real requirements” (de Carteret & Vidgen, 1995 p.368). The solution then lies in improving “the engineering process such that the requirements specification is accurate, complete, consistent and unambiguous” (*ibid.* p.368). From an objectivist viewpoint, not only is a ‘correct’ data model the ultimate goal but also its quality can be empirically tested by its correspondence to reality. Lakoff (1987) has questioned the value of adopting this position in all situations, writing

“In objectivist semantics, the world simply is the way it is, and truth cannot be affected by the way one understands a situation...Objectivist semantics assumes that in any domain there is only one correct way to understand what

---

<sup>2</sup> A much fuller discussion of the behaviour that can be expected from modellers working within each of the four paradigms can be found in de Carteret and Vidgen (1995)

is going on. It is one thing to make such a claim for physical sciences. It is quite a different matter to make such a claim for social or abstract domains, where alternative models may be equally valid” (p.201).

If reality is viewed as being socially constructed then data modelling becomes a matter of seeking consensus rather than uncovering ‘truth’ and is not so much an activity of discovery but of negotiation (Veryard, 1994). De Carteret and Vidgen (1995) describe this subjectivist position as one in which,

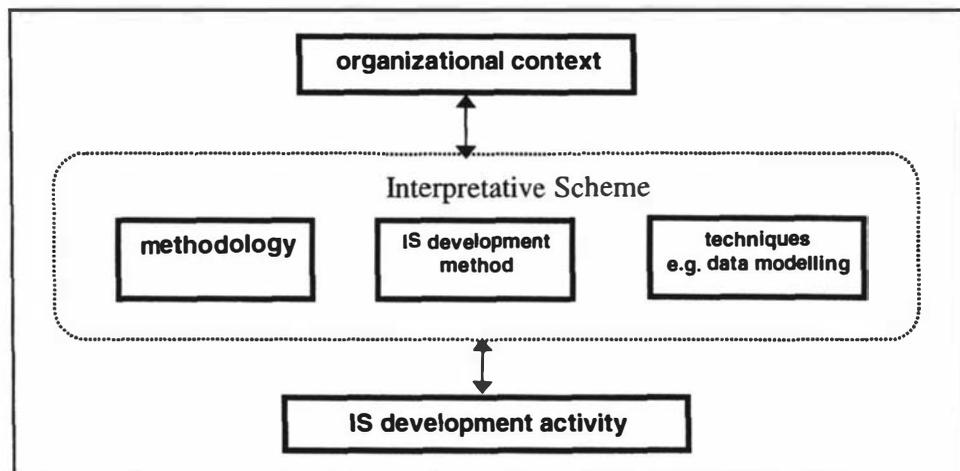
“there is no single correct data model - an acceptable model will emerge through the process of constructing a data model. The more people whose interests are reflected in the model then the more likely it is that a shared understanding will be reached and a successful implementation achieved. A good data model is one which creates a shared understanding; the appropriate way to go about creating such a model is through participation and facilitation” (p.368).

Modelling behaviour in this instance will typically involve a number of ‘stakeholders’ ranging from the business specialists, to the program designers and database administrators whose views need to be integrated (Darke & Shanks, 1994a) into a workable design from which a useful database can be constructed. As Simsion (1994) observes, “ support for the model by all stakeholders..is critical” (p.116). This position then actively encourages and expects modellers to find alternative model solutions and to have a reliable means of choosing between them. Thus, the suitability or quality of a model can only be decided within the context for which it was built and is based on necessarily subjective judgements.

De Carteret and Vidgen (1995) point out that by insisting that data modelling is either objectivist or subjectivist we create a “binary opposition between (them) and cast them as a dualism” (p.370). They suggest that the fuzzy approach (Kosko, 1993) of viewing them as a duality rather than a dualism is more appropriate, and ultimately more useful. Their interpretative scheme is illustrated at Figure 4.

In their view both objective and subjective aspects are seen to be present in all elements of the IS development process at the same time. They argue that it is “not appropriate to consider the process as objective (IS development mirrors organisational reality) or as subjective (organisational reality is created through IS development)” but to recognise that the potential for both exists (de Carteret & Vidgen, 1995 p.372). In this view an objectivist modeller will be likely to mirror and thus reinforce the existing

organisational structures. A modeller operating from a subjectivist standpoint, on the other hand, will tend to re-interpret and re-create these structures in different, and possibly innovative, ways. Their 'interpretative scheme' can be viewed as a bridge between the complex multi-realities of the 'real world' and the singular reality of the electronic database.<sup>3</sup>



**Figure 4. IS Development as a duality (de Carteret & Vidgen, 1995)**

As so often, Kent (1978) foreshadowed this argument, commenting that, "although it (*a conceptual model*) is a single perception of reality it must be broad and universal enough to be transformable into the perceptions of all the applications supported by the data base" (p.28). It may be well that the tendency, noted earlier, to equate conceptual data modelling with the modelling of users' information requirements, is a pragmatic attempt to describe this very phenomenon.

The interpretative scheme of de Carteret and Vidgen (1995) also provides an interesting framework in which to view the apparent inconsistencies and contradictions that emerge in both academic and practitioner literature. A significant amount of academic research has concentrated on providing richer, more complex and more formal Models with which to better capture the 'objectivist' reality. Consequently, experimental research

<sup>3</sup> Although even this could now be challenged. The data warehouse is hardly a singular electronic reality bringing together as it does a vast collection of sometimes unrelated, sometimes redundant and sometimes conflicting and inconsistent data and data structures (meta data) - it begins to have many of the properties of the 'real world'.

results have often been assessed on participants' ability to reproduce the 'correct' model as previously created by the researchers themselves. Educators too, whether through textbooks or assessment measures, tend to behave, and exhort others to behave, as if there is only one 'correct' representation. Practitioners, however, have adopted a far more pragmatic approach. The existence of a number of acceptable solutions is both acknowledged (Simsion, 1994) and expected (Simsion & Shanks, 1993). Motivated by the recognition that their ultimate purpose is to design an implementable database, practitioners often choose to largely ignore those aspects of the conceptual modelling facilities which, while allowing them to be more expressive in their description of reality, provide little assistance in database creation (Hitchman, 1995).

### **Infological vs. datalogical**

The various purposes ascribed to the conceptual model seem to fit quite comfortably with de Carteret and Vidgen's framework described above. The ISO/TC97/SC5/WG3 (1982) report identified two principal purposes of the conceptual schema, firstly to describe the UoD, i.e. to provide an 'abstraction of reality' (Tsichritzis & Lochovsky, 1982) and secondly to control the descriptions in the data base; i.e. to prescribe the major data structures that will be implemented (van Griethuyzen, 1983). The first purpose implies that the conceptual schema, formulated independently of DBMS considerations, should be understandable to UoD experts. However, there is no clear indication as to whether this abstraction is intended to represent a singular or a consensual reality. The second purpose requires the representation to be a reflection of the potential database structures and clearly acknowledges an element of knowledge construction. While recognising these two distinct purposes the report does not recognise any tension between them nor any specific difficulties in accommodating both within one representation. However, as Shoval (1985) points out, there is a clear "distinction between (the) two levels in that schema, each of which is dedicated to one of the purposes"(p.417).

Tsichritzis and Lochovsky (1982) recognise that the first purpose, which they term 'infological', is primarily used as an aid to understanding and thus validation. The second, 'datalogical' purpose, on the other hand requires a very detailed and technically oriented representation. Much research has sought to provide modelling facilities that reconcile these very different purposes. However, as Tsichritzis and Lochovsky (1982)

observe, provided that there is a means of mapping from one form of the representation to the other, then there is no reason why both purposes should be fulfilled by the same formal representation. Later researchers such as Elmasri *et al.* (1985) agree. Nevertheless, the E-R Model continues to be generally used to satisfy both needs, although not with equal success (Elmasri *et al.*, 1985) and consequently, despite the initial euphoria, has been gaining an increasingly unsatisfactory reputation ever since. To extend Simsion's (1994) architect analogy, data modellers are attempting to describe both the detailed design specifications and the artist's impression with the same representation.

It would seem that, initially at least, the E-R Model fulfilled the datalogical role adequately and in the infological role partly filled a difficult gap. However, even though conceptual data models are still usually built only as a precursor to database design, increasing levels of abstraction and an increased breadth of scope have made it increasingly important that non-IS specialists are involved in their construction. The users and sponsors of the system must understand and be able to verify the data structures and business rules (Veryard, 1994) while auditors may use the conceptual schema as an indication of database integrity (Amer, 1993). As Campbell (1992) points out the data modelling diagrams are being used for a number of purposes, which include "as a **user communication** tool (...documenting the data needs of the business) and ...as part of Systems Documentation (the ultimate purpose of which is to **communicate requirements and design**)" (p.12). It is clear that because of this role as a communication tool, the infological representation is becoming increasingly important (e.g. Shanks, 1997; Shoval & Frumermann, 1994; Siau *et al.*, 1995; Tjoa & Berger, 1993) and is often associated with research focussed on either improving the E-R Model or finding a better alternative (Blaha *et al.*, 1988; Campbell, 1992; Coad & Yourdon, 1991; Elmasri *et al.*, 1985; Hammer & McLeod, 1981; Moody & Osianlis, 1996; Schenck & Wilson, 1994; Smith & Smith, 1977a; Teorey *et al.*, 1986).

In practice, however, this infological role is not always explicitly recognised. This allows Campbell (1992,p.13) to accuse IS practitioners of "technological arrogance" in expecting users to become familiar with, and think of their data in, the constrained structures that the use of any modelling formalism, based on datalogical requirements, must use (Kent, 1978). Campbell would appear to be justified in his comment that "there is a conflict between the desire to address design issues and the need to create a form of data model

with which the business user is comfortable” (Campbell, 1992 p.13). As Olle (1993) observes, the conceptual schema must be represented in a form that is “assimilatable by subject area experts who are not familiar with informatics oriented representation forms”(p.53).

There is some consensus in the literature that the conceptual model should be,

- understood by the subject area experts whose enterprise reality it is intended to represent, (Kim & March, 1995),
- a precise, formal and complete specification of the information requirements (Tsichritzis & Lochovsky, 1982;Yunker, 1993),
- free of architectural or implementation bias (Olle, 1993; Ram 1995), and
- transformable into a logical design for a specific implementation (Elmasri *et al.*, 1985;Teorey *et al.*, 1986;Tsichritzis & Lochovsky, 1982).

However, the datalogical function of the model almost always appears to be regarded as paramount and most writers would agree with Pletch (1989) that the conceptual model is the “focus of the **database design**<sup>4</sup> process” (p.74). After all, conceptual data models are rarely built unless there is an ultimate goal of creating some slice of virtual reality, i.e. an electronic database (Tsichritzis & Lochovsky, 1982).

Despite the fact that the conceptual model is often defined in terms of its role as a “communication medium between professional analysts/designers and users” (Shoval & Frumermann, 1994 p.28), it seems that in practice it is almost always built to fulfil its primary, datalogical role (de Carteret & Vidgen, 1995). In addition, there is “little awareness of the disparity between the users’ way of thinking and the analyst’s way of modelling” (Eden, 1996 p.42). Nevertheless, it seems generally agreed that a useful infological model needs to reflect very closely the way users view data. However, users may have their own idiosyncratic ways of viewing their data (Raymond *et al.*, 1989) and these may differ from person to person and none of the views may be useful in terms of building a database. These considerations have led some researchers to investigate using some form of controlled natural language as the basis for the infological

---

<sup>4</sup> Emphasis added

representation<sup>5</sup>. After all, as Tjoa and Berger (1993) observe, “potential users of information systems usually express their system requirements in natural language” (p. 206). Natural language also provides the only way in which the “connection between a data base and the reality about which statements are to be represented, can be established” (Biller & Neumold, 1978 p.11).

### **Analysis vs. Design**

There is one further dichotomy, which it is useful to explore since it is directly related to the previous discussions. This is the tension between modelling as an analysis or as a design activity. Simsion (1994), echoing the position of many practitioners, is of the opinion that data modelling is very much a design activity and he highlights the distinctions inherent in the use of phrases such as ‘data analysis’ and database design.

In professional practice, data modellers are often called data analysts but rarely data designers; they may occasionally be called database analysts (although this is also used to refer to those who analyse the data in the database rather than its structure) and more commonly database designers. These terms, which usually cover very similar job descriptions, would seem to indicate something of an identity crisis inadvertently highlighted by Barden’s (1994) description of ‘design analysts’ and ‘database design analysts’.

Academics, on the other hand, almost always use the broad expression of data modeller and thus side step any issues of behavioural difference, although Cerpa (1995) for example, sees the process as entirely one of design. Even where the two activities are clearly acknowledged, there is still a tendency to amalgamate their characteristics. Shoval and Frumermann (1994) for example, state clearly that “data modeling is an analysis and design activity”(p.28) but later refer to “the analyst/designer” who “interacts with the user in order to understand the data structure of the reality being modeled”(p.28). Benyon (1997), while always referring to the modeller as either an analyst or a designer nevertheless uses the terms interchangeably and Burmeister (1995)

---

<sup>5</sup> Further discussion of the use of natural language within the database design process can be found in Chapter 6.

describes “the design task [*as consisting*] of two main phases - analysis of (and modelling) of the data and analysis (and modelling) the processes”(p.3). A cursory study of current textbooks (Burch, 1992; Gibson & Hughes, 1994; Hawryszkiewicz, 1997) on information systems development reveals a similar ambiguity.

Hawryszkiewicz (1997), for example, guarantees confusion by explaining,

“Data modeling is part of the development process. In the linear development cycle, it is used during the system requirements phase to construct the data component of the analysis model. This model represents the major data objects and the relationships between them. It should not be confused with data analysis, which takes place in the systems design phase. System design organizes data into good shape. Usually this means removing redundancies, a process often called normalization...” (p.182).

In case the situation is still not clear, he continues,

“Most designers develop only the high-level conceptual model in the system specification phase. **The more detailed analysis using normalization is carried out during design**<sup>6</sup> (*ibid.* p.182).

Burch (1992), too, while suggesting that analysis reports should include specific user requirements defined in terms that everyone can understand, nevertheless defines the difference between the data analysis and conceptual data design deliverables as being the level of detail represented in the E-R model. There is an inherent contradiction in this position that is never properly addressed, i.e. in order to be understandable to users the E-R model has to be simplified or abstracted to a high level, but in order to be a complete and specific statement of user requirements it needs to be detailed. Gibson and Hughes (1994) present a similar scenario although they do address the contradiction. They state that while user requirements are initially gathered during the analysis phase, where they are documented with an E-R representation, they are “refined as the logical design becomes more detailed” (p.347). They recognise that the data requirements as specified at the end of the analysis phase are incomplete and are, therefore, able to observe that “a set of satisfactory user requirements is derived through long but critically important hours of presenting **design**<sup>7</sup> documentation to users” (*ibid.* p.347). In this way they acknowledge that users are expected to confirm not just that

---

<sup>6</sup> Emphasis added.

<sup>7</sup> Emphasis added

their requirements have been adequately recorded and understood but that the detailed data structures that have been designed to maintain their information are also appropriate. It seems little wonder that there is an increased perception that users have difficulty in interacting with E-R models.

Avison and Fitzgerald (1995) confirm the exemplary nature of the above texts by equating data modelling with the analysis of “the structure and meaning of data in the organisation” (p.67) and suggesting that “entity modelling is the main technique used to achieve this in many methodologies” (p.127). They perpetuate the contradiction by describing data analysis as “an art or craft [and] not an exact science” and acknowledging that “there can be a number of useful data models” (p.128). Even in the Multiview<sup>8</sup> methodology, the data is analysed via the development of an entity model, achieved by the problem-solver extracting and naming entities and establishing the relationships between the entities (Avison & Wood-Harper, 1990). Halpin (1995) also views the analysis and design stages as overlapping and sees the NIAM-CSDP as useful in both phases.

It is clear, as Larman (1998) comments, that the “division between analysis and design is fuzzy; analysis and design work exists on a continuum and different practitioners of ‘analysis and design’ methods classify an activity at varying points on the continuum” (p.14). He concludes that “since different people and methods mean a variety of things by these terms, debating the definition is not particularly constructive” (*ibid.* p 16) and it is not helpful to be too “rigid about what constitutes an analysis versus a design step” (*ibid.* p.14). However, the issue is deeper than merely finding a convenient label. He finally concedes that,

“[n]evertheless, some consistent distinction is useful in practice between investigation (analysis) and solution (design) because it is advantageous to have a well-defined step that emphasises an inquiry of what the problem is before diving in to how create a solution. It is also sets [sic] an expectation of suitable behaviour among the team members; for example, during analysis members expect to emphasize *understanding* of the problem while deferring issues which relate to the solution, performance and so on” (*ibid.* p.16)

---

<sup>8</sup> This methodology developed by Avison and Wood-Harper (1995), is a conscious attempt to meld ‘hard’ and ‘soft’ approaches to system development by incorporating a number of techniques developed by Checkland (e.g. Checkland & Scholes, 1990).

There is also a significant difference between the philosophical positions required of an analyst and a designer that puts this discussion clearly back into the subjective/objective context discussed above. As Simsion (1994) comments,

“in analysis, creativity suggests interference with the facts. No honest accountant wants to be called creative. On the other hand, creativity in design is highly valued” (Simsion, 1994 p.7).

In other words analysts need to **behave** as if there is an objective reality awaiting their discovery and their task is to uncover this pre-existent information; designers do not. Analysis is thus best served by what Kepner (1996) refers to as the ‘rational mode of thinking’ whereby a conclusion is reached based on observed facts using reason and logic. This ‘scientific thinking’, based on immediately experienced reality, is considered the most reliable and valid. In this situation an auditable, prescriptive approach may be very appropriate.

Conversely designers are rewarded for providing elegant re-interpretations and combinations of ‘realities’, particularly if their designs provide new insights into the problem. It calls on two other modes of thinking, intuitive and creative. Kepner (1996) describes the former as,

“the thinking that occurs...when an idea simply wells up from the unconscious mind in response to the perception of a problem or issue. Information for intuitive thinking comes from the integration of stored fragments, facts and impressions that have accumulated over the years” (p.3).

and the latter as,

“thinking that reaches out beyond what is now known into what could be. It puts known elements together to form new ideas and visions. It draws on observation, experience knowledge and the indefinable ability each person has to arrange common elements into new patterns” (*ibid.* p.3).

The less prescriptive the approach, the more likely it is that innovative and creative solutions can be found. De Carteret and Vidgen (1995) clearly describe data modelling as inherently creative, where a modeller “often needs to **dream up** several different possibilities”<sup>9</sup> (p.334). They comment that “learning to be imaginative is probably much harder for the average adult than learning data modelling techniques” (*ibid.* p.334)

---

<sup>9</sup>Emphasis added

and suggest that the “tools of the trade should be used to capture creative thoughts and not to suppress them” (*ibid.* p.335). Intuitive thought is also recognised as being an important component of ‘expert’ modellers’ behaviour, particularly in their propensity to re-use previous, successful, patterns. (Batra & Davies, 1992, Simsion & Shanks, 1993)

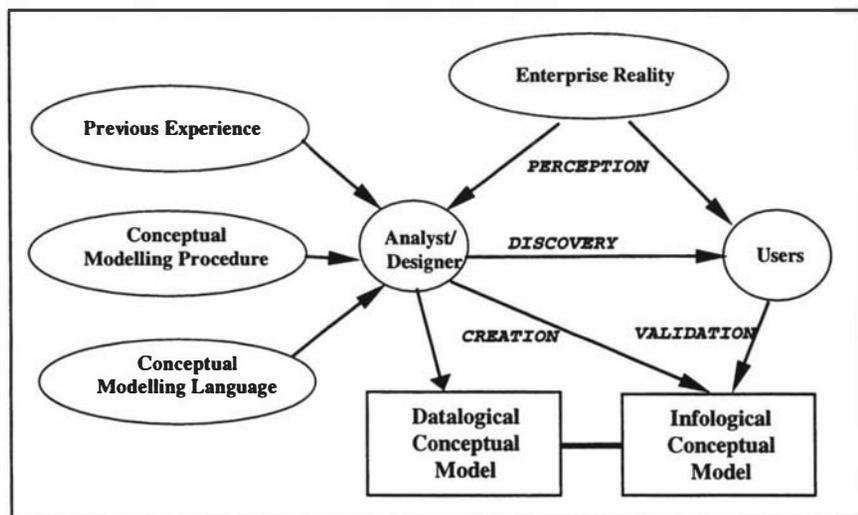
However, not everyone agrees with these characteristics. For example, Olle *et al.* (1991) classify data modelling as an analysis activity, within a framework that characterises analysis as descriptive and design as prescriptive. This would seem a reflection of the characteristics of the general E-R approach itself rather than a description of the properties required by analysis and design tools. The E-R approach can be described as highly descriptive<sup>10</sup> in its initial stages becomingly increasingly more prescriptive as it moves towards the transformation of the conceptual model into a relational implementation. Halpin (1993a) however, considers that “E-R models can be of use once the design process is finished” as they are “less suitable for formulating, transforming or evolving a design”(p.1). Thus, it would seem that, within the Information Systems community, there are some very different views on the nature of analysis and design. For the purposes of this study, analysis will be defined as an activity that seeks to determine the “elements or components of something complex” and to discover the “general principles underlying [*these*] concrete phenomena” (Brown, 1993). Design, in line with the Shorter Oxford English Dictionary, will be considered to be the action of creating a plan or picture, “in accordance with appropriate functional or aesthetic criteria...for the construction or production of a building, machine etc” (Brown, 1993). In colloquial terms then, analysis will be considered to refer to the ‘what’; i.e., to the data requirements underlying the required system development, and design to the ‘how’; i.e., to the development of possible storage structures suitable for electronic implementation.

With these definitions in mind, de Carteret and Vidgen’s (1995) framework can be seen to require elements of both analysis and design, as does the view of data modelling activities, shown at Figure 5. Certain parts of the process, specifically the requirements

---

<sup>10</sup> This issue is discussed further in Chapters 5 and 0.

elicitation phase, will require the data modeller to take on the 'analyst' role in order to record accurately each element of the users' perceived reality, and thus create a faithful map of the external territory. The result of this phase, the infological conceptual model, is most useful when it is expressed in a form that the users can readily understand and validate. It is also important that modellers are aware of their own perceptions of the UoD and do not pollute the users' view with their own interpretations and assumptions.



**Figure 5. Conceptual Modelling Activity Kim & March (1995) adapted.**

However, in the next phase, the creation of the datalogical conceptual model, the data modeller needs to behave as a designer, not an analyst, to create an elegant resolution of the possibly conflicting viewpoints that have been uncovered. At this point, the modeller is a designer, responsible for creating a data structure that will satisfy the diverse requirements of the users of the eventual system, as well as providing for re-use and flexibility. This is not likely to be attained by a mechanistic mapping of the various viewpoints into a superview alone. It is likely to require some 'flair' that comes directly from the data modeller's own experience and expertise. At this stage creativity is at a premium and it is here that data modelling becomes very much an art. As Simson and Shanks (1993) observe "implicit in the design view of modelling is the assumption that most practical data modelling problems have many workable solutions"(p.3).

One further consideration is that there must be an auditable correspondence between the documented results of the analysis phase and the data structures that are the result of the design activity. This is represented on the diagram at Figure 5, by the bold line linking the infological and datalogical models.

## Summary

Olle (1993) appears correct then in observing that “since the term ‘conceptual schema’ was introduced, it has come to mean different things to different people” (p.49). Conceptual data models may be considered as a reflection of reality, as a database map, as a record of a portion of the one ‘real’ universe or as an agreed consolidation of a number of subjective views of the world. Additionally, they may be intended to enhance communication about the world or to facilitate the design of an electronic data storage system and they may be seen to require, either or both, analysis or design skills and behaviours. Almost any combination of these views may be held by researchers, educators and practitioners and it is not unusual for the combination to appear contradictory as de Carteret and Vidgen (1995) point out.

One suggested view of conceptual data modelling is that it is part of an interpretative scheme whereby a portion of the world, as it is perceived by any number of users, is transformed into an appropriate datalogical design (de Carteret & Vidgen, 1995). The process has two quite distinct functions, both of which may benefit from having their own specific representational forms. The first, infological, function would be well served by being “based on constructs as close as possible to the human way of perceiving information” (Kent, 1978 p.195) and the second, datalogical, purpose could be filled by the use of the relevant implementation model, e.g. the Relational Model (Elmasri *et al.*, 1985). However, it must be possible to map directly and consistently between them (Tsichritzis & Lochovsky, 1982). Modelling behaviour and the choice of an appropriate modelling tool are undoubtedly affected by the standpoint from which all these aspects of conceptual data modelling are considered. Some interesting and significant issues relate to being able to clearly differentiate the purpose and outputs of the two stages and to the question of whether it is appropriate to use the same tools, techniques and methods to undertake both activities.

For the purposes of this study then, a conceptual model is considered to have two roles, which, while generally viewed in combination, may be better, served by different representations. The first role, as a communication tool, is to provide a “precise/formal specification of the information which the user wishes to communicate to and from the proposed information system...which should be free of architectural or implementation

bias” (Yunker, 1993 p.3). It is the result of an analysis of the users’ UoD and consequently must be easily understood by the users in order to enable accurate verification. As Larman (1998) points out, “[a] critical quality to appreciate about a conceptual model is that is a representation of real-world things not of software components” (p.87). However this model must also provide the basis for the data design by being “transformable to both the required user views and to a variety of data storage and access structures” (Sharp, 1993 p.2). The more straightforward the transformation, the more accurate the mapping of user requirements to the final design, will be. This model will be referred to as the ‘infological’ or ‘analysis’ conceptual model.

The second role is to provide a representation of data structures that would support the users’ information requirements in the paradigm of the eventual DBMS. As such it is a product of a design activity, which proposes structures that could be constructed for a specific implementation technology. Such a representation must be understood and meaningful to other IS professionals such as systems analysts and database designers but while it must be possible to verify that the users’ requirements are still being supported, it should not be necessary for a user to comprehend technical design structures. There is thus a clear need to extract from this model a view of user requirements, which is comprehensible to the users and which is preferably similar in form to their initial verified view. This model will be referred to as the ‘datalogical’ or ‘design’ conceptual model.

This study will argue that the infological and datalogical roles of the conceptual model can benefit not only from having different representational forms but also by recognising the different behaviours required by modellers during the process of constructing them. In addition, it will take the position described by Avison and Fitzgerald, accepting that the infological model, as a partial reflection of reality, can only ever be one, subjective view, neither “absolutely correct or wrong, but more or less useful” (Larman, 1998 p.98). However, this study will also accept that the datalogical model, as a map of potential or actual implemented data structures, can be viewed as an objective representation of an electronic database. The process of modelling will thus be viewed as an interpretivist activity, in line with the arguments put forward by de Carteret and Vidgen (1995).

# 5 E-R Modelling: observations

*“Philosophers will always wonder what an entity is. Meanwhile there is some information processing to do!” (Brodie et al., 1984 p.3)*

## Introduction

Although the use of object-oriented techniques is increasing, some version of the original E-R Model (Chen, 1976), using some variation of the methodology proposed by Teorey *et al.* (1986), is still a widely used conceptual data modelling technique (Avison & Fitzgerald, 1995; Bock & Ryan, 1993; Siau *et al.*, 1996; Flynn, 1998). It is a central part of many system development methodologies, ranging from SSADM4 (Eva, 1994) to Information Engineering (Finkelstein, 1989) and is widely supported by many CASE tools. There is evidence that in New Zealand the percentage of organisations employing formal data analysis techniques for information systems development, while low, is growing and that many of these organisations are using CASE tools which support some variation of E-R modelling (MacDonell, 1994). It also seems likely that the E-R Model “is here to stay” (Loosely & Gane, 1990 p.7), at least for the immediate future.

Even the development of object-oriented modelling techniques does not always move us very far from the E-R tradition. Yourdon’s Object-Oriented models, for example, bear a marked similarity to the conventions of E-R diagrams (Coad & Yourdon, 1991) and Benyon-Davies (1992a) suggests that the inherent strengths of E-R modelling can be extended in such a way as to provide a useful and effective object model. Blaha *et al.* (1988) go even further claiming that “[o]f the many approaches to relational database design, the Object Modeling Technique is particularly effective”(p.414). A claim which is perhaps partly explained by their insistence that, in OMT, “[t]he notion of an object is synonymous with entity in the ER and LRDM methods” (*ibid.*p.416) and is further illuminated by their observation that “[t]hird normal form is an intrinsic benefit of object modelling” (*ibid.* p.418).

## Criticisms of the E-R Model

Many of the earlier claims, that the E-R Model provides an easily conceptualised model (Yao *et al.*, 1984) using constructs that are highly intuitive (Brodie *et al.*, 1984), and useable as an effective communication tool (Teorey *et al.*, 1986), are now being questioned (Goldstein & Storey, 1990; Hitchman, 1995). Yet despite the diverse criticisms, it remains a common choice for both high level corporate modelling as well as database design. Perhaps the answer lies in the fact that “higher levels of abstraction such as E-R diagrams are conducive to creative thinking and effective communication” (Blaha *et al.*, 1988 p.415) and that the production of E-R models is seen as an essential component of the design strategy. Nevertheless it is criticised for being difficult to use, teach and understand. The reasons for this apparent contradiction may lie in three areas,

- that the E-R Model is no longer used in the way and for the purpose for which it was first proposed,
- that the E-R Model referred to by practitioners and some academics is not the E-R Model at all, and
- the necessarily descriptive method employed to construct an E-R model.

The first of these areas has been investigated in previous chapters and will not be discussed in detail here. However, it is worth re-iterating that E-R models were initially used to represent the conceptual schema and were thus essentially datalogical in nature. Consequently, they were generally used by those professionals who had already acquired a competent understanding of the physical structure of databases. As a diagrammatic tool for representing this complex abstraction (the database) they were, no doubt, ‘intuitive’ to many of their early users. However, as an increasing number of non-technical users have been required to validate their semantic content, E-R models have not been found to be necessarily intuitive to those people who are used to thinking of their data in some other form.

Kent (1978) has suggested that much of the difficulty that a user experiences in learning to interpret E-R models has less to do with the complexity of the tool than with the struggle, “to contrive some way of fitting his problem to the tool: changing the way he thinks about his information, experimenting with different ways of representing it”. He continues, “much of this “learning” process is really a conditioning of his perceptions so that he learns to accept as fact those assumptions needed to make the theory work, and

to ignore or reject as trivial those cases where the theory fails” (Kent, 1978, p.194). This is supported by Goldstein and Storey (1990) who suggest that users need to have a good understanding of the E-R Model before data models become useful to them. Additionally, Kim and March (1995) believe that the users in their experiment may have scored more highly in discrepancy checking tasks with the E-R model than had been anticipated because they had a “higher-than-expected degree of record-orientation<sup>1</sup>” (p.110). This finding, while supporting Goldstein and Storey’s (1990) proposal, goes further in suggesting that the greater the users’ understanding of physical database structures, the more able they are to interpret and validate the E-R model correctly. Kim and March (1995) conclude that as users become increasingly familiar with E-R models and presumably as their ‘record-orientation’ increases, the quality of the conceptual data models will improve. It seems likely that this problem has become more noticeable in recent years, due in part to the large increase in the number of people involved. However, it may also be due to a phenomenon that has crept insidiously into the practice of data modelling but gone largely unremarked by either practitioners or academic researchers; the development of the E-R/Relational hybrid mentioned previously<sup>2</sup>.

### **The E-R/Relational Hybrid**

When the E-R model was originally proposed, it was intended to express data requirements in a way that was independent of the underlying implementations. Initially then, it made little sense to incorporate any aspects of the classic Models into an E-R model; for example, it can be argued that normalising an E-R model is not necessarily useful unless the target DBMS is relational. Some writers such as Avison and Fitzgerald (1995) who, despite relating normalisation to the Relational Model, still consider it to be applicable in other data structuring contexts, challenge this view. However, it is interesting that neither Chen in 1976 nor Teorey *et al.* in 1986 viewed normalisation as a part of constructing an E-R Model. Indeed, in the LDMRD (Logical Design Methodology for Relational Databases), Teorey *et al.* (1986) specifically describe normalisation as a step that follows the mapping of the E-R model to the candidate relations of the Relational

---

<sup>1</sup> This term appears to signify familiarity with physical data storage, possibly relational, structures

<sup>2</sup> See Page 31

model. However, as relational DBMSs increasingly became the only target, so a number of relational considerations found their way into the E-R conventions. The result is that many so-called E-R models can be better described as relational models using E-R graphical notation and certainly a number of CASE tools use a subset of E-R conventions<sup>3</sup> to provide diagramming facilities for the construction of a normalised relational schema (Ryder, 1993). Some practitioners even go so far as to dismiss the need for a conceptual model at all. Huff (1992), for example, suggests that,

“It is both unrealistic and unnecessary to design data structures independent of the type of the target DBMS (unrealistic because of problems of physical implementation, unnecessary because DBMSs in companies usually live longer than conceptual data structures)”(p.33).

There is evidence in the literature that would seem to confirm the existence of this unacknowledged hybrid. Marche (1993) refers to data models, especially normalised entity relationship ones” (p.44). Cerpa (1995) too, considers that “a conceptual model is an entity-relationship model<sup>4</sup> in 3NF<sup>5</sup>” (p.353), while Batra and Zanakis (1994) specifically state that an “E-R diagram developed with the relational representation as the target must obey the normalisation concepts” (p.228). Benyon (1997) goes further and declares that “a completed E-R model consists of a diagram (the E-R diagram), a corresponding set of fully normalised (i.e. in 5NF) relations and additional information” (p.160).

However, if an E-R model has to conform to normalisation rules, it cannot claim to be a user’s view of the world “without any consideration of how data is stored on computerised files” (Jarvenpaa & Machesky, 1989 p.367). Normalisation is a property of relations (Cerpa, 1995) and is, therefore, a relational process that relies on the existence of well defined, minimal primary keys, and the transformation of information bearing relationships into entities, including the resolution of many to many relationships. These latter considerations are appropriate to the design of relational databases but not necessarily to those of other implementation paradigms and, with the exception of primary keys, were not a part of the original E-R Model (Chen, 1976).

---

<sup>3</sup>The relationship diamond is almost never used.

<sup>4</sup> This is referenced to Chen1976.

<sup>5</sup> This definition is constrained by the acknowledgment that it is “for the purpose of this study”.

An E-R model that is fully normalised is thus not really an E-R model, as envisaged by Chen, but a relational model represented by similar graphics. This distinction may go some way to explaining some of the contradictory research. After all a relational model, masquerading as an E-R model, may well be more difficult for users to understand, providing, as it does, a much more constrained view of their data than a 'purer' E-R model would require. Like Hitchman (1995), Kim and March (1995) make the interesting comment that a possible explanation for the lack of understanding of some of the semantic concepts that they observed, may be attributable to the lack of such constructs in the Relational Model. They also go on to say that the findings do not necessarily show that the respondents could not produce relational data structures (Kim & March, 1995). As Simsion (1994) remarks, "from a practical perspective, there is not much value in adopting a data modelling language that is not compatible with current database management systems or CASE products"(p.22). Indeed, this would seem to reflect the situation that Hitchman (1995) found among practitioners in the UK, that "a significant number...do not seem to understand some available semantic constructs of the entity-relationship model" (p.39).

The existence of the hybrid is hinted at by Batra *et al* (1990) who observe that "in the general database design literature and in **the practice of professional data analysts**<sup>6</sup>, the Relational Model has been used to conceptualise data requirements" (p.126) and also that some form of E-R graphical notation is commonly used to depict these relational models. There is little direct evidence of how widely the hybrid is used. However, personal observation and the findings of some researchers suggest that practitioners are using the E-R conventions and terming the products of this usage E-R models, although they are actually constructing implementation-oriented, relational designs rather than communication-oriented, conceptual models (Hitchman, 1995; Kim & March, 1995). Simsion and Shanks (1993) for example discovered that among 39 practising data modellers in Australia not one used the Chen diamond notation for relationships. This may help to explain Kim and March's (1995) conclusion that "users as well as IS analysts should be trained in an appropriate conceptual modeling formalism" (p.111), as well as explaining why the E-R Model would appear to be no longer seen as a particularly useful

---

<sup>6</sup> Emphasis added

communication tool (Hitchman, 1995). A clear expression of this confusion among practitioners is highlighted by Chee-Pun Wong (1995), in the practitioner magazine 'Database Programming and Design', where it is stated (incorrectly) that "E-R modeling...does not support attributes within relationships" (p.42). The statement however, would be quite legitimate if referring to relational modelling, which is of course the actual formalism under discussion.

The merging of E-R graphical notation with the construction of logical, relational schemas is common in textbooks and training manuals, many of which are adopting a pragmatic attitude towards the hybrid, perhaps giving an indication of the extent of the current practice. Simsion (1994) in a chapter entitled 'The Entity Relationship Approach' describes how to represent a previously normalised relational design as an E-R diagram; while in a chapter entitled 'The Conceptual Schema' Avison (1992) specifically details the normalisation of the entities in an E-R model. Benyon (1997) is quite specific:

"there is a very close correspondence between the E-R model and the relational model. Indeed, in the methodology provided in this text, together they form the information model...The E-R model being the main user-oriented part of the information model and the relational representation being the computer oriented part" (p.180).

De Carteret and Vidgen, on the other hand, make no reference to the E-R model and are quite clear that they are writing about relational modelling. For them "another way of representing an entity is as a relational table" and they are careful to refer to diagrammatic representations as "Entity Diagrams" rather than the more common 'Entity-Relationship diagrams' (de Carteret & Vidgen, 1995 p.12). However, this is unusual. Most texts either describe clearly a purist E-R modelling exercise and then its transformation into a 'logical' relational design (e.g. Ricardo, 1990; Sanders, 1995; Teorey & Fry, 1982) or compress the two phases into one data modelling activity where the E-R/Relational hybrid is clearly evident (e.g. Benyon-Davies, 1996; Simsion, 1994; Veryard, 1984; Veryard, 1994; Watson, 1996). The anonymous authors of the AURISA Data Modelling Workshop in 1991 were quite clear about their rather contradictory position,

"The modelling techniques covered in this workshop are based on E-R modelling...E-R models are ...independent of the physical data model under which the database will be implemented. The approach adopted by the

authors however, is to develop models from which relational structures are **directly derivable**<sup>7</sup>“ (p.9).

There may be another, very pragmatic, reason for the widespread adoption of this hybrid model by practitioners and this relates to the actual process of creating a useful model. A truly infological E-R model may not be directly or easily, transformable into a relational schema. A lack of appropriate unique identifiers, the existence of one-to-one or many-to-many relationships, the existence of derived data and relationships that carry attributes, may accurately reflect the users' view of their domain. However, it may also force the modeller to make decisions, based either on a personal interpretation of the enterprise or on previous experience, at the point of transformation to a datalogical representation<sup>8</sup>. It may well be that the better a conceptual E-R representation is at fulfilling its infological role, the less effective it is at its datalogical role and vice versa. If conceptual models are mainly constructed by IS practitioners rather than users (Hitchman, 1995), then it would not be surprising that the resultant models were favouring the relational, datalogical representation. It may also be, for reasons that will be explored later<sup>9</sup>, that the construction of a datalogical representation is significantly easier, particularly for less experienced modellers who also have an understanding of relational theory.

It is clear that the E-R/R hybrid models are being constructed and are commonly used in professional practice and the benefits of this approach must seem attractive to developers working within strict budget and time constraints. By combining the infological and datalogical functions of the conceptual model into one representation, there are apparent savings<sup>10</sup> in seeming to speed up the database design process and facilitating the design of the logical data structures on which the functional analysis is often based. However, there are a number of negative implications of this cost cutting. As has been previously suggested, users are reportedly finding it difficult to understand and thus validate these representations. This must reduce the models' effectiveness as

---

<sup>7</sup> Emphasis added

<sup>8</sup> Or even earlier. Benyon (1997) suggests that in the E-R model, M:M relationships should be decomposed during the early stages of data analysis.

<sup>9</sup> See Page 66

<sup>10</sup> For example, it becomes unnecessary to build two models and to maintain a mapping between them.

strategic planning and communication tools, as well as being of little use in recording enterprise rules. If even those users who have been closely involved with the construction of the model find difficulty in interpreting it, it is unlikely to be of much use in a wider arena or over a period of time. The models are unlikely to use elements of the E-R Model, which, while providing rich semantics are not readily implementable in a relational database. Finally, while organisations may believe that they are constructing a 'conceptual' model for their applications or even for their enterprise, they are unlikely to reap the future benefits implied by the meta-data architecture previously described<sup>11</sup>.

### **Constructing an E-R/Relational hybrid model**

There is another factor relevant to the development of the hybrid and this relates specifically to the **process** by which it is constructed. While it is recognised that conceptual modelling is a complex activity, particularly for non-expert designers, there has been little research on the actual process of conceptual model construction (Batra & Srinivasan, 1992; Yunker, 1993; Benyon-Davies, 1992b; Srinivasan & Te'eni, 1995) particularly on the activities within the discovery phase (Kim & March, 1995). Batra and Zanakis (1994) point out, that while most techniques provide methods to translate an E-R diagram to a relational schema, "they do not detail a precise set of rules and heuristics to develop the E-R diagram itself" (p.228). Eden (1996) too comments that "the literature falls short of providing a definitive and comprehensive account of the process of data analysis by the Entity-Relationship approach" (p.42). The guidance generally given is summarised by Firms (1993) thus,

"One approach is to develop the diagram first and then to define data elements and assign these as attributes to the entities in the data model and finally to specify key structures and other details. The second approach is to define the data elements first, then develop the diagram and assign attributes to entities as the diagram evolves. In practice a combination of the two approaches provides the most intuitive basis" (p.115).

While these instructions are specifically related to the use of a CASE tool, the general principles mirror those in most descriptions of the process. Some authors provide a case study to illustrate the process or more commonly rely on simple straightforward examples.

---

<sup>11</sup> See Page 35

Eden (1996) identifies that, with the exception of NIAM, none of the literature, “succinctly addresses the problem of how to securely guide the novice from a requirements statement in Natural Language to a conceptual schema” and few “recognise the importance of linguistic analysis” (p.43). Instead, novices are directed to identify the ‘important’ entities and the relationships between them.

Beyond a broad definition of what an entity is, little direction is given in how to identify them. Interviews with users, analysis of input and output documents are usually recommended starting points but apart from pointing out that the nouns uncovered in these studies are useful pointers to possible entities the modellers are left to determine from their own intuition and experience which nouns will make useful entities. Interestingly, one of the most succinct renditions of this advice comes from an object-oriented textbook where Eaglestone and Ridley (1998) observe,

“most design methods are informal in the way in which the task of identifying candidate objects is achieved, and provide only heuristics, i.e., rules of thumb, to guide the designer. A common approach is to analyse natural language descriptions of the organisation which the database is to serve and select candidate objects on the basis of grammatical clues. In general, an object is implied by a noun or noun phrase” (p.284).

Thus, ‘experience is the best teacher’ seems to be the maxim and the process remains a necessarily subjective one. Simsion (1994) summarises the situation by saying that “entity identification is essentially a process of classifying data, and there is considerable room for choice and creativity in selecting the most useful classification” (p.82). It is, therefore, no surprise that 51 data modellers of varying experience, given exactly the same problem description, will come up with 51 different data models (Simsion & Shanks, 1993), nor that Bubenko (1986) views the quality of a conceptual model as being totally dependant on the competence of the modeller. To the objectivist, of course, there can only be one ‘correct’ way of categorising the data but practice suggests that regardless of the validity, or not, of this view, it is not useful. Traditional objectivist views of classification and categorisation have been challenged (Lakoff, 1987; Way, 1991) and this debate continues to inform research on the conceptual modelling process. However, it has not yet resulted in any significant guidelines for the apprentice modeller. In addition, it has been observed that as physical electronic databases require strict typing the conceptual modelling facilities have tended to incorporate them as well (Tsichritzis & Lochovsky, 1982).

Entity identification, or the categorisation of data into useful sets, would appear initially to be a fairly straightforward task and certainly the attention paid to the activity in most textbooks would seem to confirm this impression. However, as Raymond *et al.* (1989) point out, considering that, “there are more than 11 billion ways of choosing four categories of five items from a set of 20” (p.239), the task is obviously not a trivial one, particularly as many of these possible combinations will not be suitable for any task. A number of writers comment on the difficulty of identifying entities (Benyon-Davies, 1992b; Flavin, 1981; Teorey *et al.*, 1986; Veryard, 1994) while others comment on the even greater difficulty experienced by novice modellers in determining the relationships between them (Batra & Sein, 1994; Shoval & Frumermann, 1994). While novices could easily gain the impression that a set of pre-existent entities and relationships are awaiting their discovery, some writers make it clear that there is definitely “more than one workable answer in most practical situations” (Simsion, 1994 p.6) and that part of the modelling task is to identify information structures that are ‘useful’ to the enterprise.

The need to use the categories in some way; e.g. to retrieve members of the categories at some point according to some criteria, is thus likely to determine the usefulness of the categories that are chosen. Shoval and Frumermann (1994) extend this observation to the identification of objects, by saying, “there is no standard or “normal form” that can guide an OO designer to decide which representation is preferable and much depends on the assumptions about the types of queries that will be posed at run time” (p.31). Despite an earlier report suggesting that the “organizations in which subjects chose to express information are in large part a function of the relationships among the data, and only secondarily a function of subjects’ preferences” (Durdin *et al.*, 1977 p.8), Raymond *et al.* (1989) designed an experiment partly around the belief that the subjects’ “effectiveness in solving the retrieval task would reflect the effectiveness of their structuring” (p.241). Now, if the implicit target implementation for most conceptual models is relational then it is highly likely that the most useful categories in an E-R model will be perceived to be partly normalised, candidate relations. After all, there would appear to be little to be gained in undertaking the categorisation task twice. If a modeller knows that the categories will have to obey the normalisation rules eventually, there must seem little point in not doing so from the outset. However, to use this criterion for category creation is immediately to move away from the language and structure of the users’ perception of

their enterprise reality and into the realm of computer storage design. Thus, even with the stated intention of creating an E-R model, it is easy for modellers to be seduced into creating instead an essentially relational model in which the E-R is, very clearly, just a thin layer on top of the basic relational model (Date, 1995).

Unfortunately without the assistance of the normal forms, entity identification becomes even more difficult and subjective. Relying on vague definitions of an entity as “something that exists and is distinguishable” (Blaha *et al.*, 1988 p.415), or the slightly more definitive “a thing or object, whether real or imagined, about which information needs to be known or held” (Barker, 1990 p.G1-3), is not very productive, as data modelling students soon discover. Not only does confusion arise as to whether an entity is an entity at all but it is compounded by the realisation that an entity may be viewed as a relationship or as an attribute depending on the context or the modeller’s own preference (Date, 1995). With no clear guidelines on these issues it certainly seems safer to equate entities with relations, as Benyon (1997) specifically does, disallow relationships that contain attributes and use normal forms to assist in determining the appropriateness of the resulting structures. Expert practitioners may well exhibit this behaviour and certainly the practically oriented books would seem to reflect such a situation (e.g. Benyon, 1997; de Carteret & Vidgen, 1995; Simsion, 1994; Veryard, 1984).

This, then, would help to explain both the findings mentioned earlier (Goldstein & Storey, 1990; Hitchman, 1995; Kim & March, 1995) and why instruction in data modelling is often preceded by an overview of relational databases or a description of normalisation (Avison, 1992; Simsion, 1994; AURISA, 1991). Indeed, de Carteret and Vidgen (1995 p.xi) specifically state, “our experience of training data modellers has shown that it is easier for a student to understand the idea of entities and relationships at the conceptual level having first understood how they might be implemented.” A modeller then seems to need to have an understanding of relational databases in order to build a useful E-R model. This is a clear indication that the model required is not only a datalogical one but specifically a relational one. Teorey *et al.* (1986) appear to have implicitly recognised this in providing “guidelines for classifying entities and attributes [*that*] will help the designer converge to a normalised relationship [sic] design” (p.204), while Batra and Zanakis (1994) specifically use relational considerations in developing heuristics for their conceptual database design approach.

## Alternative Approaches

There have been some attempts to provide an alternative approach to the process of requirements discovery and entity identification. As early as 1983, Kent (1983 p.3) claimed that “deferring problems of identification and representation until later in the process” resulted in a method that was “at once more simpler and more powerful than other methodologies.” He claimed that his proposed method did not “describe procedures for identifying the entities and relationships” although later he contradicts this assertion suggesting that “records are chosen in an objective and systematic way based largely on the pattern of...relationships in the facts to be maintained” (*ibid.* p.30). He highlights that choosing the “major entities” in a system is not always obvious and that it can even, “invert the design process: [*as*] users develop an ability to choose major entities that will correspond to the records they want in the data design” (*ibid.* p.30). In the same year, Chen (1983) was also exploring ways of extracting entity relationship models more directly from the natural language description of the UoD but the purpose was still essentially datalogical and the ideas appear to have been largely ignored. More recently Eden (1996 p.42) has proposed a method which he claims is a “step by step process for developing a conceptual schema from natural language requirements” having recognised the need for novices to have “a well defined process”. However, his process still requires the identification of “strong entities and attributes” as the first step and thus does nothing to alleviate the problems of this initial stage.

Currently there appears to be increasing dissatisfaction with the E-R Model as a tool for conceptualising data requirements. This has led academic researchers, many of whom persist in repeating uncritically that ‘one of the most widespread conceptual data modelling methods is the E-R model first proposed by Chen’, (e.g. Avison & Fitzgerald, 1995; Bock & Ryan, 1993; Siau *et al.*, 1995), to believe that it is the ‘pure’ E-R Model that is under fire. However, it is more likely to be the E-R/Relational hybrid that is both the most ubiquitous Model and the target of much of the more recent criticism. Consequently, researchers continue to search for more rigorous Models or develop richer semantic constructs for existing ones (e.g. Codd, 1990).

Practitioners too, experiment with using other methods, currently object-oriented ones, that are considered to reflect reality more naturally than the models in traditional

analysis (Martin & Odell, 1992) despite the difficulties of mapping from such an object model to a relational database design (Olds, 1997). Nevertheless, the relational, datalogical considerations are never far away, with practitioners 'normalising' object models (Martindale, 1997) and Blaha *et al.* (1988 p.414) stating that "object modeling promotes adherence to normal forms". Meanwhile, both groups largely fail to acknowledge that the practitioners' 'E-R Model' is fundamentally relational, and that many of the criticisms might best be answered by complementing it with an additional, appropriate infological representation rather than seeking to replace it.

### Summary

Flavin remarked in 1981 that "[c]urrent practices of analyzing business systems for the construction of databases are largely intuitive and undisciplined" (p.9) and it seems that little has changed. As Marche (1993) observes "the theoretical foundations upon which data modelling rests are remarkably incomplete when considered critically. The principles for building a data model as set out in the literature are contradictory, unclear, unspecific and/or incomplete" (p.44); this observation has significant implications not only for the education but also for the practice of Information Systems professionals. Additionally there is an expectation that the data models that are built by this *ad hoc*, descriptive process should be able to adequately fulfil two important, complex and essentially different functions. The failure of an E-R model to do this is seen as a failure of the Model, the modellers or the users, depending on the critic's own perspective but never as an indication that it is the expectation that is, at fault.

Returning once again to the architect analogy, it is as if the builders, the architect and the prospective owners and users of a new building were asked to accept as adequate a plan that was an uncomfortable mixture of building specifications and artist's impression. No doubt the non-technical users would find it difficult to confirm whether what they were being shown was what they wanted, the builders would have difficulty in knowing exactly what to build and the architects would consequently find themselves in the uncomfortable and pivotal role of attempting to assure both sides of the satisfactory nature of this hybrid. Accepting the architect's word would require a significant degree of trust in both his integrity and professional expertise. Avison and Fitzgerald (1995) make exactly this point,

“In information systems development there is no clear-cut distinction between artist’s impressions and the engineer’s blueprints. There is not one version of the model for the user and another version for the computer programmer. Some may argue that this may be a valid goal, but furthering our analogy, artist’s impressions are notoriously optimistic and vague about difficulties, and engineer’s blueprints are very difficult to interpret by all but the trained. It is not satisfactory for the untrained to have to accept the statement ‘trust us, we’re the experts’.

This means that the users and the builder of the information system must both understand the conceptual model<sup>12</sup>” (p.123).

However, Avison and Fitzgerald (1995 p.33) note elsewhere, the unsatisfactory nature of this arrangement, commenting that the documents which users are required to ‘sign-off’ as correct are “completed by computer-oriented people” and are not designed for users. It is unlikely that this situation would be acceptable to any of the stakeholders in the process of commissioning, designing or constructing a major building. Additional conceptual aids, such as scale models or photographs of existing buildings of a similar nature, are used to substantiate artists’ impressions. With the appropriate adaptations, why should the construction of a database be any different?

---

<sup>12</sup> The term ‘conceptual model’ is used here in the context of Soft Systems Methodology and is not intended to describe any form of data model. However, the sentiments seem to be equally valid for both meanings.

# 6 NIAM: observations

*“By focusing on the facts to be maintained in a data base, we obtain a methodology for data analysis and design which is at once simpler and more powerful than other methodologies.” (Kent, 1983 p.3)*

## Introduction

As the need for business users to participate in the construction of effective and useful data models has become recognised, one area of research that has developed is the exploration of the possibilities offered by natural language to provide this communication bridge (Métais *et al.*, 1993; Sowa, 1984; Sowa, 1991; Steinberg *et al.*, 1994; Tjoa & Berger, 1993; Vadera & Meziane, 1994; Way, 1991). However, Object-Role Modelling (ORM), particularly as used within NIAM and as supported by the CASE tool, *InfoModeler*<sup>TM</sup>,<sup>1</sup> already provides a natural language interface between the user and the modeller. ORM also fulfils Tsichritzis and Lochovsky’s (1982) mapping requirement, in that the conceptual model it produces can be transformed by a straightforward algorithm (Nijssen & Halpin, 1989) into a normalised relational schema. NIAM is widely used in Europe and Australia and is increasingly recognised as one of the major data modelling approaches (Creasy, 1989; Kim & March, 1995; Laender & Flynn, 1992; Song & Forbes, 1991; Weber & Zhang, 1991).

## Object-Role Modelling

Originating in the early 1970’s, ORM views the world as made up of objects playing roles (Halpin, 1995) and “traditionally expresses all information in terms of elementary facts, constraints and derivation rules” (Halpin, 1993b p.1). There have been several

---

<sup>1</sup> *InfoModeler*<sup>TM</sup> is the registered trademark of Asymetrix Corporation. All references in this study are to *InfoModeler*<sup>TM</sup>, 1.5 Desktop version

methodologies developed for the creation of an ORM, of which NIAM (Natural Language Information Analysis Method) is the best known. "The fundamental approach of building a design by *starting with specific examples* and thereafter following a well-defined procedure" (Nijssen & Halpin, 1989 p.31), was initially developed by Nijssen during his work at Control Data in The Netherlands. Nijssen and Halpin (1989 p.31) attribute the initial proposal "to base conceptual schema concepts on elementary natural language sentences" to Falkenberg (1976) whose approach was in turn influenced by the work of the linguist Fillmore (1968). NIAM has subsequently been independently developed by both Nijssen (1994) and Halpin (1995).

Hitchman (1995) reports that Nijssen has demonstrated that ORM and the E-R Model are capable of expressing a similar level of meaning, a view supported by Laender and Flynn (1994). Bronts *et al* (1995) too, conclude, "there is little difference in the way of modelling of E-R and NIAM" although in their terms the "way of working" is significantly different<sup>2</sup>(p.232). Apart from anecdotal evidence (e.g. Halpin & Orlowska, 1992), however, there is nothing to show whether the fact-oriented approach is more or less effective. Recent comparative studies of E-R and NIAM are inconclusive (Laender & Flynn, 1994; Kim & March, 1995; Shoval & Even-Chaime, 1987) although Weber and Zhang (1991) conclude that the constructs provided by NIAM are more powerful than those provided by the E-R Model. Halpin (1995) suggests that ORM and E-R modelling may have a complementary use although his suggestion is limited to using E-R diagrams as a convenient means of summarising complex ORM models. However, even this limited suggestion is unusual with most advocates of ORM maintaining the method's superiority over the E-R Model with almost religious fervour. It is not the intention of this study to enter this debate.

## **NIAM-CSDP**

The complete NIAM method (termed the NIAM-ISDM) is made up of three stages, conceptual schema design (the NIAM-CSDP), conceptual schema transformations and relational implementation. A summary of the seven steps that make up the CSDP is

---

<sup>2</sup> This idea is discussed further in the following chapter.

shown in Figure 6 (Halpin, 1995 p.43). In essence, the steps provide a clear and well-defined procedure for building the conceptual schema, by capturing information requirements as natural language sentences, termed ‘facts’, extracting sentence patterns or ‘fact types’ from these sentences and using ‘real’ examples to validate the facts and assist in identifying the required constraints. On completion of the CSDP, the resulting schema can be adjusted in the conceptual schema transformation stage although any generated alternatives must be equivalent in meaning to the original. This schema is then transformed into a normalised, relational schema by a straightforward published algorithm (Nijssen & Halpin, 1989).

1. Transform familiar information examples into elementary facts, and apply quality checks.
2. Draw the fact types and apply a population check.
3. Check for entity types that should be combined and note any arithmetic derivations.
4. Add uniqueness constraints for each fact type and check arity of fact types.
5. Add mandatory role constraints and check for logical derivations.
6. Add value, set comparison and subtyping constraints.
7. Add other constraints and perform final checks.

**Figure 6. The 7 steps of the NIAM-CSDP (Halpin, 1995)**

While some aspects of the process can be tedious and time-consuming when undertaken manually, the use of a CASE tool such as *InfoModeler*<sup>TM</sup> not only speeds up but also significantly simplifies the later stages of the process. With *InfoModeler*<sup>TM</sup>, once the ‘facts’ have been added to the model, together with their constraints and example values, the tool is able to determine and create the diagrammatic representation of the sentence, including the basic constraints of uniqueness, cardinality and optionality. It is also able to validate the model for syntactic correctness, transform the ORM model into a database schema in “optimal normal form<sup>3</sup>” (Nijssen & Halpin, 1989, p.254) and generate appropriate SQL commands for a variety of DBMSs. It is, therefore, theoretically possible to create a physical schema from an entered set of natural language sentences, without further intervention, although in most cases some refinement is desirable.

---

<sup>3</sup> Optimal Normal Form or ONF is described as basically equivalent to 5th Normal Form, although the “number of 5NF tables in the overall schema has been minimized” for efficiency. (Nijssen and Halpin, 1989 p 254).

Details of all the phases of the NIAM-ISDM can be found in Nijssen and Halpin (1989), while Halpin (1995) has produced an extended version of NIAM, termed FORM (Formal Object-Role Modelling) which provides the underlying paradigm for the *InfoModeler*<sup>TM</sup> CASE tool. This study is mainly interested in the first step of the CSDP process, i.e. the collection of information examples and their transformation into elementary facts and fact types. This step, identified by Halpin and Orlowska (1992) as not only the most important but also “the foundation of NIAM’s design procedure” (p97), is procedurally equivalent to the identification of entities and relationships in the E-R approach.

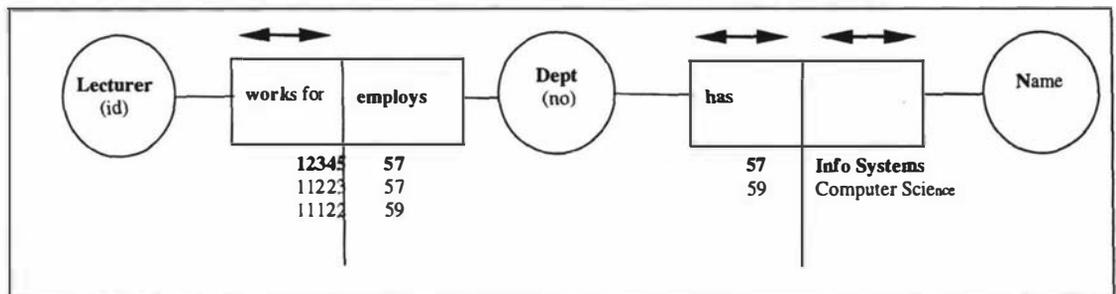
### Natural Language

NIAM theory begins from the axiom that all information communicated can be expressed as a set of elementary declarative natural language sentences from which general patterns or ‘fact types’, as shown in Figure 7, can be extracted (Nijssen, 1994). NIAM has its foundation in “linguistic theory and applies set theoretical concepts to induce formal information grammars from (these) sets of natural language sentences” (Schouten, 1993 p.1). Two further axioms state that all communication with the user is held in the user’s language exclusively and that all communication with the user is illustrated with practical examples. This is justified by the argument that users are most comfortable describing their enterprise in their own natural language and that with a representative set of example sentences they are able to assess the validity of the set, allowing even complex constraints to be determined (Nijssen, 1994). As Biller and Neuhold (1978) remark, “it is very important to rely on the users understanding of natural languages, since only in this fashion can the connection between a data base and the reality about which statements are to be represented, be established” (p.11).

<p><b>F1 Department (number) employs staff member (id)</b></p> <p>Department with number “57” employs staff member with id “1122”          Department with number “57” employs staff member with id “2233”          Department with number “59” employs staff member with id “3344”</p> <p><b>F2 Department (number) has name (value)</b></p> <p>Department with number “57” has the name “Information Systems”          Department with number “59” has the name “Computer Science”</p>
--

**Figure 7. A set of example sentences for two qualified fact types**

NIAM also provides a graphical notation for representing the objects, roles and constraints identified by the elementary sentences and a simple example is provided at Figure 8. However, with the exception of some of the more complex constraints, which are more easily shown on a diagram, the natural language ‘elementary facts’ and their diagrammatic representation are equivalent in the sense that the graphical notation can be automatically transformed into the ‘elementary fact types’ or *vice versa*. This equivalence, sometimes termed ‘semantic equivalence’ is perhaps better described by the expression ‘data equivalence’ utilised by Biller and Neuhold (1978) who provide an informal definition of it in asserting that, “two data bases are equivalent if they represent equivalent facts about a certain slice of reality”(p.12). However, they are clear that in using this term they are assuming “that it is known, whether two sets of natural language sentences are equivalent. Again we presuppose that the natural language is commonly understood” (*ibid.* p.12).



**Figure 8. A simple NIAM diagram**

InfoModeler™ capitalises on this perceived ‘data equivalence’ by providing a ‘Verbalizer’ report which displays the graphical representation of each ‘elementary fact type’ and its natural language equivalent together with the natural language examples which have been entered. This report, illustrated in Figure 10 on page 96, provides a version of the conceptual schema considered suitable for user verification.

### **The ‘Elementary Fact’ Concept**

The most fundamental concept in NIAM is that of the ‘elementary fact’, derived from familiar, concrete examples within the UoD. An ‘elementary fact’ is defined as an assertion that an object plays a role or that one or more objects participate in a relationship (Nijssen & Halpin, 1989). In other words, it is an assertion about the UoD. The choice of the term ‘fact’ is not incidental but indicates that the system is to treat the assertion as

being true of the UoD whether or not this is actually the case in the ‘real’ world (Halpin, 1993a). However, as Halpin (1993b) concedes it is difficult to “define the notion (of an elementary ‘fact’) precisely”(p.2) although the definition given above is a useful working definition. Halpin also concedes that “expressing information as elementary facts is not always easy” (*ibid.* p.3) but nevertheless feels that the benefits more than justify wrestling with any difficulties. He expresses those benefits as follows: -

- “By dealing with information in simple units we stand a better chance of getting a correct picture of the application being modeled;
- Constraints are easier to express and check (e.g. all functional dependencies should appear as uniqueness constraints and because facts types are shorter the number of possible constraint patterns in each one is reduced);
- The conceptual schema is easier to modify, since fact types can be added or deleted one at a time, rather than modifying compound fact types<sup>4</sup>;
- The same conceptual schema can be used to map to different data models (if we group fact types together into compound fact types on the conceptual schema, different groupings may actually be required in some target data models)” (*ibid.* p.3).

Halpin (1995) makes no mention here of the benefits of interacting with the users in their own natural language, benefits which are seen as self-evident by the NIAM community.

Some forms of ORM insist that all ‘elementary facts’ are binary, however, NIAM allows elementary facts to be of any ‘arity’, i.e. unary, binary, ternary or higher, although binary facts are certainly the most common. Halpin (1993b) has demonstrated that all NIAM elementary facts of whatever arity can be expressed in binary form but usually at the expense of natural expressiveness. From the statement ‘*Hedgehogs hibernate*’ for example, NIAM would allow expression of the unary fact:

(1) ‘The animal with type ‘hedgehog’, hibernates ‘

which, as a binary fact, would need to be expressed in a form such as:

(2) ‘The animal with type ‘hedgehog’ has HibernationStatus of ‘H’.

Both these sentences qualify within NIAM as legitimate ‘elementary facts’ but as a major concern of NIAM is to “bridge the semantic gap between the informal user world and the formal modelling world” (Yunker, 1993 p.15), Halpin (1995) prefers to retain

---

<sup>4</sup> He is referring here to an E-R type ‘entity’ as a compound fact type.

the first, more natural, form. Indeed, InfoModeler™, utilising the optimal normal form transformation algorithm, will map both sentences into the same relational structure (Appendix 1).

Determining whether or not a 'fact' is elementary, can be problematic. In general terms, a 'fact' is elementary if it cannot be broken into two or more 'facts' without losing information. For example, *Hedgehogs live in England*, rendered as: -

(3) The animal of type 'hedgehog' lives in the country with name 'England', is a legitimate 'binary fact'. It cannot be expressed as two simpler sentences without losing the information that hedgehogs live in England. However, the sentence *Hedgehogs live in England and hibernate*, rendered as: -

(4) The animal of type 'hedgehog' lives in the country with name 'England' and hibernates,

can clearly be broken down into the two 'elementary facts' (1) and (3)<sup>5</sup>. However, the distinction is not always so obvious. It is possible to find some linguistic heuristics to assist the modeller in judging when a 'fact' is elementary. The use of the conjunction 'and', for example, can often provide a clue that two 'elementary facts' are within the one sentence. However, this guideline is by no means foolproof as sentences (5) and (6) illustrate. Sentence (5) has no 'and' but is clearly intended to convey the same semantics as sentence (4). Sentence (6) on the other hand is a legitimate quaternary 'elementary fact', which despite the use of the conjunction cannot be further decomposed without information loss.

(5) The hibernating animal of type 'hedgehog' lives in the country with the name 'England'.

(6) The student with the student identifier '95000012' enrolled in the Paper with the code '57.366' and obtained a Grade of 'B' in the Year '1997'.

In these circumstances, modellers are required to determine the 'elementarity' of a 'fact', by reference to the known constraints provided by the concrete examples (Halpin, 1993b) although the final arbiter in unclear cases must always be the domain expert (Collingnon & van der Weide, 1994).

---

<sup>5</sup> Of course this does not follow if hedgehogs that live in England hibernate but those resident elsewhere, do not!

## The Construction of Elementary Facts

There are three clear stages in the construction of 'elementary facts' (van der Lek *et al.*, 1992). The first is the collection of concrete examples from the UoD, which serve to illustrate the relevant information. Within NIAM, it is recognised as being the users' responsibility to provide these examples, which are usually taken from both the input and output documents of the system and from interviews with the users themselves. It is critically important to the entire CSDP that the set of examples is sufficiently rich to describe all possible facts about the UoD (Calway & Sykes, 1995). The assumption underlying this, that the domain experts can and should provide a complete set of significant examples (Collingnon & van der Weide, 1994) has been criticised by some as being very limiting (Darke & Shanks, 1995c). Certainly the only guidelines that are given to meet a situation where suitable examples are not available, e.g. for a new system, are rather unsatisfactory. In this situation the analyst is advised to "begin by getting the user to write down some examples, and then work from these" (Nijssen & Halpin, 1989 p.35). As Darke and Shanks (1995c) also point out, there are other aspects of requirements elicitation and definition that are not addressed adequately in this stage. These are the social context in which the activity is taking place, and the resolution of potential conflict that can arise over either the problem definition or alternative viewpoints of the information requirements. These criticisms will be discussed further in the next chapter.

The second stage of the process is the verbalisation, or expression in natural language sentences, of the examples. This verbalisation is comprised of a set of sentences describing all the objects in the UoD and the roles that they play. For example, sentences derived from an example listing of employee details could include verbalisations such as,

A. Adams has the employee number '715' and works in the Sales Department. His office is in room 2.23 and his phone number is 4206...C. Smith has the employee number '716' and works in the Finance Department. His office is room 3.21 and his phone number is '4242'...

This verbalisation is merely an intermediate step to provide a natural language basis from which the 'elementary facts', such as

The Employee with the number '715' has the name 'A Adams'  
The Employee with the number '715' works in the 'Sales Department'

can be derived. In practice, most experienced modellers will often move directly from the examples to 'qualified facts' from which the fact types can be derived, in much the same

way as an experienced relational modeller will often instinctively create entities in third normal form. ‘Qualified elementary facts’ have a formal structure depending on their ‘arity’. Every ‘qualified fact’ must have a minimum of one object, reference mode, label and predicate as this defines a ‘unary fact’, i.e.

<object>, <reference mode>, <label>, <predicate>

For ‘binary facts’ the first three elements are repeated after the predicate, thus,

<object>, <reference mode>, <label>, <predicate> <object>, <reference mode>, <label>.

The *objects* are the things of interest, the *reference mode* is the property of the object which allows one to identify which instance of the object is being referred to, the *label* is the actual value and the *predicate* is the role that the object(s) are participating in. Some of the ‘qualified facts’ from the previous verbalisation could be expressed thus: -

The EMPLOYEE with **employee #** ‘715’ has the NAME (**with the value**) of ‘Adams A’  
 The EMPLOYEE with **employee #** ‘716’ has the NAME (**with the value**) of ‘Smith C’  
 The EMPLOYEE with **employee #** ‘715’ works for  
     the DEPARTMENT with the **name** ‘Sales’  
 The EMPLOYEE with **employee #** ‘716’ works for  
     the DEPARTMENT with the **name** ‘Marketing’

Here the objects are shown in upper case letters, the reference modes in bold, the labels are italicised and the predicates are underlined<sup>6</sup>. These facts are often written in abbreviated form with the superfluous words, such as ‘with’, ‘the’ and ‘of’, omitted and the reference mode shown in parentheses e.g.

EMPLOYEE (**employee #**) ‘715’ has NAME (**value**) ‘Adams A’  
 EMPLOYEE (**employee #**) ‘715’ works for DEPARTMENT (**name**) ‘Sales’

NIAM insists that each ‘qualified fact’ must include at least one object, which must participate in at least one role<sup>7</sup>. Additionally the reference mode and label of each entity type object<sup>8</sup> must also be expressed. Thus the hibernating hedgehog of Sentence (1) provides an example of the minimum expression allowable in NIAM, in the form,

ANIMAL (**type**) ‘hedgehog’ hibernates.

<sup>6</sup> The convention shown here differs slightly from Halpin’s. He suggests that value types are not given a reference mode, instead the name of the value type is followed directly by the label.

<sup>7</sup> However FORM allows for the inclusion of “lazy entities” i.e. an object (or more correctly entity) that exists without participating in any fact, i.e. has no predicate (Halpin, 1995 p.164)

<sup>8</sup> Entity or entity type is used to differentiate between those objects which are ‘described’ by some property, i.e. they are not just value objects.

## Elementary Fact Types

The final stage is the extraction of the elementary ‘fact types’ themselves. Noticeable patterns emerge in the ‘facts’ as illustrated above and the ‘fact types’ are the expressions of these general patterns. By removing the label, a number of identical sentences are left e.g.,

Fact 1. EMPLOYEE (**employee #**) has NAME (**value**)  
 EMPLOYEE (**employee #**) has NAME (**value**)

Fact 2. EMPLOYEE (**employee #**) works for DEPARTMENT (**name**)  
 EMPLOYEE (**employee #**) works for DEPARTMENT (**name**).

With the redundant sentences removed, these are the generalised ‘fact types’ which provide the structure of the conceptual schema and which can be graphically recorded. The construction of these ‘fact types’ marks the end of the first step of the CSDP. However, the use of the concrete examples, as depicted within the elementary ‘facts’, does not end here. In Step 2 they are used to provide entries into the ‘fact table’ as shown at Figure 8 on page 75. In Step 4 the user is asked to judge whether certain examples are permitted or not in the information base. The results of these enquires are used to identify the uniqueness constraints for each fact type which are then shown as a double headed arrow over the column or columns in the fact table which are required to have unique values.

## Summary

The NIAM-CSDP process is thus significantly prescriptive, even in the discovery stage of requirements analysis. The procedure sets out not only the steps that must be undertaken in order to specify a formal model but also how to express the grammar using the specific fact encoding constructs (Yunker, 1993). This prescription is claimed as providing a consistent, reproducible process and a verifiable resulting grammar (Schouten, 1993; Yunker, 1993), both of which are considered by Yunker (1993) to be essential pre-requisites for a “true engineering discipline” and necessary goals for conceptual data modelling (Yunker, 1993 p.4). Interestingly it also appears to have been one of the goals of Teorey *et al.*’s (1986) method for E-R construction, which is described as producing “nearly reproducible designs from a given requirements specification.” (p.220)

NIAM appears to display some balance between the infological and datalogical roles of the conceptual model. The elementary sentences act as a natural way of expressing a user's requirements, while the examples of those sentences provide both a straightforward way of determining the basic constraints and an aid in minimising ambiguity. Although the sentences can be represented diagrammatically it is not usually necessary for the user to view them in graphical form. Even where this is required there will have been no artificial structural constraint placed on the information by the modelling paradigm. The model while directly transformable to a relational schema does not require any specifically relational constructs and is thus theoretically equally suitable for mapping to any implementation paradigm.

The advocates of NIAM are very clear about both its perceived benefits and its superiority to the E-R approach. For example, Schouten (1993) expresses the opinion that NIAM is "unchallenged as a reliable information analysis method" (p.1). Leung and Nijssen (1988) make the point that the use of natural language as the conceptual modelling tool minimises the loss of semantic information in the earliest and most critical stage of development. Yunker (1993) suggests that NIAM is inherently superior as a method as it provides a prescriptive, auditable procedure, something, which, he argues, is precluded in the E-R approach by the nature of the modelling constructs that are provided<sup>9</sup>. Although many of these arguments are persuasive and well founded and while the shortcomings of the E-R Model are well documented and generally agreed, nevertheless the E-R approach remains the technique of choice for many of the organisations which choose to undertake data modelling. This choice is possibly a reflection of the historical lack of commercial CASE tool support, the preference of the data modellers themselves and perhaps also reflects a certain resistance to the use of NIAM among practitioners who are already skilled in the use of the E-R approach.

---

<sup>9</sup> This argument is revisited in more detail in the following chapter.



# 7 E-R and NIAM: a comparison of approach

*Data modeling focuses primarily on linguistic phenomena. This is not to deny that it is also concerned with finding efficient storage structures. This is still a significant design problem in data modeling. All data modeling however, presumes either implicitly or explicitly some form of modeling of data meaning, because it is only through the knowledge of data meaning that an understanding of the design problem...is possible." (Hirschheim et al., 1995 p.28)*

## Introduction

Several comparisons have suggested that there is little fundamental difference between the E-R Model and NIAM's OR Model (Bronts *et al*, 1995; Kim & March, 1995; Laender & Flynn, 1994). Laender and Flynn (1994), for example, conclude that, "despite the fact the two models differ significantly in terms of the surface characteristic of diagrammatic representation, they are very similar in their modelling capability" (p.255). The BWW (Bunge-Wand-Weber) model developed by Wand and Weber (1993, 1995) has been used to evaluate both NIAM (Weber & Zhang, 1991) and E-R modelling, as supported by the CASE tool, Excelerator V1.9, (Green, 1997). Weber and Zhang concluded that NIAM, "provides a rich array of constructs " that "are more powerful than those provided by the ERM (p.80) but suggest that, "NIAM will tend to be used with other methodologies that compensate for its semantic deficiencies" (*ibid.* p.81). Green (1997) reaches a similar conclusion for E-R models and he successfully predicted that users would have problems "with the recording and integrating of business rules into their designs and that such users will combine tools and/or grammars...to overcome the resulting ontological incompleteness" (p.9). Indeed, he observed that 7 of the 10 participants interviewed used free text to represent business rules.

## A further comparison

If this were the complete picture there would seem little practical purpose in making yet another comparison between the two techniques. Bronts *et al.* (1995) however, have suggested a framework of IS development methods which highlights an interesting difference between NIAM and E-R. Their framework of IS development methods is illustrated at Figure 9 and the six, boxed components are described as follows,

- 1) The *way of thinking* describes the assumptions and viewpoints of the methodology and thus makes explicit the philosophical framework in which the methodology is used.
- 2) The *way of working* both defines and orders the tasks and sub-tasks that are to be performed and also provides guidelines and heuristics on how these tasks should be carried out.
- 3) The *way of modelling* describes the constructs together with their properties and the permitted relationships between them. In other words it provides the grammar and syntax of the 'language' in which the models are to be expressed.
- 4) The *way of controlling* sets out how the use of the methodology should be managed.
- 5) The *way of supporting* details how tools, particularly CASE tools, support the methodology.
- 6) The *way of communicating* describes the form in which the models are to be communicated to human beings, usually in some form of graphical notation. Bronts *et al.* (1995) note that different methods may be "based on the same *way of modelling*, and yet use a different graphical notation" (p.214).

They suggest that the combination of the ways of *modelling* and *communicating* are "usually referred to as a modelling technique" (*ibid.* p.214). This chapter extends this concept and defines the combination of the ways of *modelling*, *communicating* and *working* as a definition for a **data modelling approach**. In doing so, it refines the definition of a data modelling approach, provided by Loosely and Gane (1990) and discussed in Chapter 3. Bronts *et al.* (1995) having studied both E-R and NIAM in the context of their framework, concur with Laender and Flynn (1994) that there is little difference between the two approaches in terms of their *way of modelling* although their *way of communicating* (e.g. their diagrammatic conventions) is significantly different. However, the main difference that they identify "lies in their respective *ways of working*" (Bronts *et al.*, 1995, p.232), although they do not describe the details of that difference.

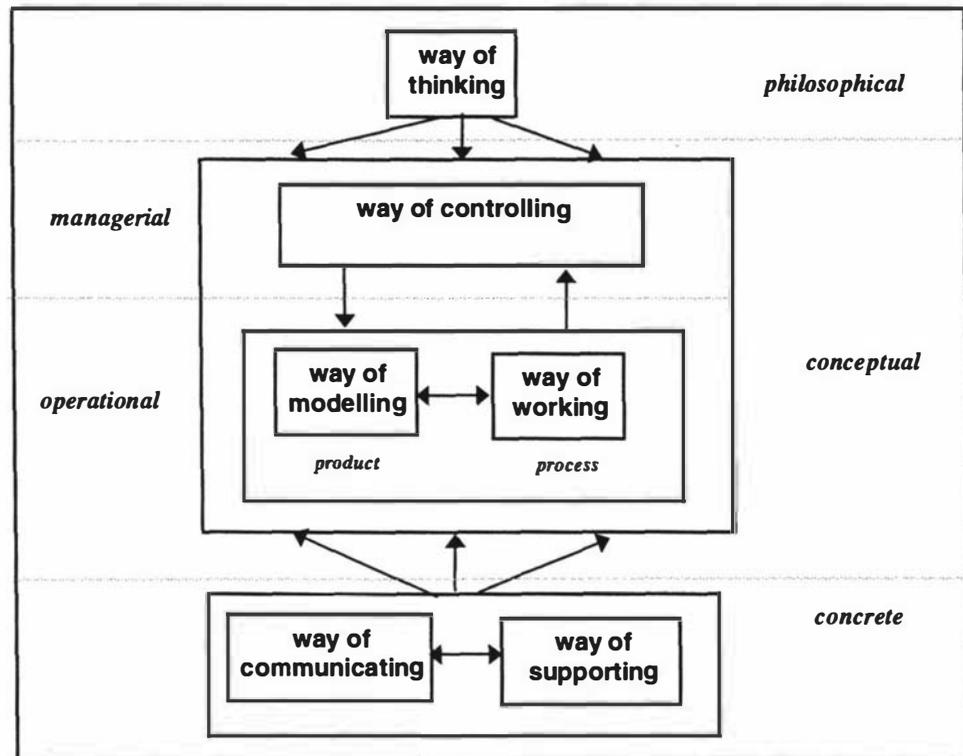


Figure 9. A framework for IS methodologies, (Bronts *et al*, 1995)

In a less rigorous analysis, Barden (1994) has suggested six major differences between NIAM and what he terms the 'traditional approaches'. These differences<sup>1</sup> are that NIAM,

- 1) provides a well defined procedure for undertaking the modelling process,
- 2) provides a way to model without requiring the modeller to classify entities or attributes, too early in the modelling process,
- 3) provides a means of validating the model for 'correctness'<sup>2</sup> as a preparation for its mapping to a relational schema,
- 4) provides a formalised natural language means of communicating a model as well as a graphical representation,
- 5) allows for the graphical representation to be populated with example values to assist in the validation of the model, and
- 6) provides for a wide range of constraints that can be mapped directly either to the relational model or even to the target DBMS.

<sup>1</sup> The ordering of these differences has been changed from the original to facilitate classification.

<sup>2</sup> Although not explicitly stated, syntactic correctness is implied.

Interestingly these differences, drawn from practitioner experience, support Bronts *et al.*'s (1995) findings by providing three differences in the way of working (1-3), two differences in the way of communicating (4-5) and one in the way of modelling (6). It is these differences, particularly 1, 2, 4 and 5, that suggest that not only can the two modelling techniques be complementary but also, as will be argued in Chapter 7, that they are inherently suited to different stages within the Information Systems development life-cycle. Accordingly these four differences will be discussed more fully. Previous chapters have sought to show that there is a major difference between the E-R approach<sup>3</sup> and NIAM-ISDM in modelling procedure, particularly in the initial stage of knowledge elicitation and discovery. The characteristics of this difference may be discussed within the framework of descriptive and prescriptive approaches of which the E-R approach may be generally described as the former and NIAM as the latter.

### **Differences in the way of working - procedure**

#### ***A descriptive approach***

A descriptive approach essentially lays out what to do but not how to do it. Examples are given of possible inputs and outputs but little direction is provided as to the appropriate process that is required, to either identify the 'usefulness' of the input or to transform it into a 'useful' output. Such an approach obviously allows for considerable individual choice and creativity and encourages intuition, innovation and the re-use of previous experience. Idiosyncratic preferences will emerge in the work of experienced practitioners while trial and error will be an inevitable element in the strategy of the less experienced. It is unlikely that others can exactly reproduce the final output, as the use of such a process is similar to the production of an art form. In essence, it is analogous to giving someone a plate of spaghetti bolognese, a cupboard of possible ingredients and some rudimentary instruction on the use of saucepans and the effect of heat on various foodstuffs. With no further guidance and little previous knowledge, a novice cook, even after close scrutiny of both the inputs and desired output is likely to take many attempts to reproduce an acceptable version of the initial dish. An experienced

---

<sup>3</sup>This encompasses the approach of constructing both an 'academic' E-R model and a 'practitioner' E-R/Relational hybrid.

cook on the other hand will have little difficulty in drawing on previous experience to create something, probably different in subtle ways, yet, possibly, superior to the original. The descriptive approach delights in diversity and innovation and implicitly acknowledges a subjective reality where alternative models may be equally valid (Lakoff, 1987) and are in fact actively encouraged (Shanks *et al.*, 1993).

Even a fairly cursory study of the guidelines given for the development of E-R models, which here include the E-R/Relational hybrid, discussed earlier, reveals the essentially descriptive nature of the E-R approach. This nature is highlighted by the method traditionally adopted in teaching its early stages. In summary, the instructions given require the modeller to identify the 'useful' entities, identify the relationships between them and finally identify the properties or attributes of the entities<sup>4</sup>. These instructions may be expanded by suggestions that nouns appearing in interview transcripts or input/output documents may provide clues as to possible entities (e.g. Veryard, 1984). However, studying and working through examples is usually the only real substitute for experience that is offered to novices. Simson and Shanks (1993) also point out that, in apparent contradiction to the essentially creative nature of a descriptive approach, "few texts offer more than one viable solution to data modelling examples" thus helping to perpetuate "the underlying assumption that there is a single 'right' answer" (p.2).

The descriptive nature of the E-R approach is also alluded to by Firms (1993) in describing data modelling as an "inherently intellectual process...dependant upon the individual...their thought processes, the methods by which they gather information...and the sequence of events by which they acquire such information" (p.114). Both Pletch (1989) and Simson (1994) see data modelling as a creative process, being described by them, respectively, as "a process of using one's imagination" (Pletch, 1989 p.76) and one where the element of choice is fundamental. Simson (1994) indeed, states quite specifically that the responsibility for identifying entities rests "squarely on the data modellers shoulders" (p.124) and that there will be many different variations of possible models.

---

<sup>4</sup> Possibly relationships too, in a Chen based E-R model

There are a number of problematic implications of a descriptive approach, some of which may well have a detrimental effect on the construction of E-R models. Such an approach requires a significant experiential learning phase leading Shanks *et al.* (1993) to conclude that novice modellers could benefit from a period of apprenticeship with an experienced data modeller. This suggestion, of course, reflects an ancient and effective pedagogical tradition for encouraging the development of talented artists and craftspeople. However, unless or until, conceptual modelling becomes recognised either as an art form or, at least, as a craft akin to haute cuisine, it seems unlikely that the commercial world will be prepared to invest in such expensive training. It could be that one of the reasons for the lack of data modelling activity noted by both Hitchman (1995) and MacDonell (1994) is that effective E-R modellers are long in the making. This would make experienced data modellers fairly rare, relatively expensive and perceived, perhaps, as something of a 'luxury'.

Another important implication of a descriptive approach is the variable quality of the output. The earlier discussion of entity identification<sup>5</sup> suggested that data categorisation is individual, arbitrary and creative. It is thus not surprising that once a model is expected to represent a domain of greater complexity than most textbook examples, a number of variations are possible. Batra and Antony (1994) reported the number of solutions produced by their subjects was "close to the number of subjects themselves" (p.63). They also observed that "different subjects viewed the situation somewhat differently" and that their models showed that their subjects understanding of the UoD "was not usually identical to the semantics conveyed by the correct solution, but overlapped to some extent" (p.63). Simsion and Shanks (1993) also found that 51 data modellers, of whom 39 were practitioners, created solutions to a complex problem with a range of between 5 and 31 entity types and where "as far as reasonably could be determined, no two models were the same or even very similar" (p.7). As Bubenko (1986) observes, the quality of the conceptual model is thus completely dependent on the competence of the modeller. With a descriptive approach it seems almost inevitable that alternative solutions will be produced.

---

<sup>5</sup> See Page 66

Despite the difficulties of evaluating the alternative representations, the need to choose between them is real and necessary. Although semantic completeness would seem to be an obvious measurement of quality, it is rarely the primary one<sup>6</sup>. Instead a conceptual model will often be evaluated in terms of its effectiveness as a basis for physical design. In other words, the datalogical function is again paramount, possibly at the expense of understandability.

A reason for this may lie in another effect of the use of a descriptive approach, the need for heuristics to aid the process. Batra and Antony (1994) identified that novice modellers, faced with complexity, generally resorted to heuristics. They concluded that the misapplication of heuristics, such as anchoring<sup>7</sup> were the cause of a significant number of errors in their subjects' models. They also concluded that anchoring is so common because of the form of the usual training methods, which usually "focuses on the strategy used to arrive at a correct answer but skirts around incorrect strategies and the possible pitfalls"(p.67). They continue,

"The fault lies with the instruction method...The challenge is to find a set of rules and heuristics that can be imparted in a limited time and can handle most situations. It would be interesting to study how experts capture their rules and heuristics...Experts frequently draw from past experiences, and whether their approach can be applied to novices is a question that can only be resolved by systematic research" (*ibid.* p.67).

It is interesting that at no point do Batra and Antony (1994) question the efficacy of the E-R approach itself or its suitability as a novice tool. Neither do they speculate on what might constitute effective heuristics although they do hint that database theory might play a useful, although not exclusive, role. Despite this hint however, they do not suggest the use of a heuristic commonly found in texts, in research articles and in practice - normalisation.

As previously discussed<sup>8</sup> one clear way of assessing the 'usefulness' of entities is to view them as candidate relations and once this step has been taken it is a much smaller one to consider normalising them. By normalising entities, novices can lessen the

---

<sup>6</sup> A full discussion of the evaluation of conceptual models appears in the next chapter.

<sup>7</sup> Anchoring is defined as the formulation of an initial hypothesis, which the subject makes little or no attempt to revise.

<sup>8</sup> See Page 64

impact of the 'outcome irrelevant learning effects', identified by Batra and Antony (1994) as a reason for the use of the anchoring heuristic. If an entity is a normalised, candidate relation then a modeller can, by applying the principles of relational theory, make a judgement as to the possible effectiveness of the structure. The conceptual model thus becomes an unacknowledged E-R/Relational hybrid model. Thus, implementation effectiveness becomes a measure of quality and users find it necessary not only to understand basic relational concepts but also to be comfortable viewing 'their' data in relational form, in order to be able to validate its semantic correctness.

### ***A prescriptive approach***

A prescriptive approach however, lays out both what to do and how to do it. Consequently, it is far less dependent either on subjective judgements of the modeller or on heuristics. As this approach allows for little, or no, individual choice or creativity, the results of the process tend to be consistent and reproducible, and the process itself is auditable. Such an approach has no need to emphasise the output, as it is assumed that the right inputs, subjected to a methodical and accurate application of the steps, will always provide a sound and predictable output. Returning to the 'spaghetti bolognese' analogy, the subject is provided with a detailed recipe which describes not only the required ingredients but the exact process for transforming the correct quantities into the desired end product. It may not be necessary for the subject to study the end product at all, or certainly not in any detail. At any stage it is possible to see which steps have been completed and an acceptable outcome is guaranteed provided that the recipe itself is proven and that the steps are followed with a high degree of accuracy. An acceptable outcome is, therefore, no longer completely dependant on the modeller, in fact there is a general assumption that most people will be capable of following the prescription, even those with little expertise or previous knowledge of the process. However, unless the subject develops an understanding and awareness of the reasons for and the effects of, the steps in the recipe and learns to adapt the process and substitute ingredients, there is no possibility of innovation or flexibility. It might be argued that such an approach is to be commended in a grill chef at a fast food outlet but not in a master chef at a top restaurant. In other words, a prescriptive approach followed, with no imaginative or thoughtful input from the modeller, will be capable of creating adequate outputs but will not facilitate innovative problem solving or produce new insights.

The steps of the NIAM-CSDP discussed in the previous chapter, are significantly more prescriptive than the E-R approach. This characteristic is emphasised by its advocates who claim, for example, that the, “*prescriptiveness* of the method guarantees that we can always reproduce the analysis results (*reproducibility*), that we can always verify the results (*accountability*) and that we can teach the method to almost everybody (*teachability*)” (Schouten, 1994 p.E2). Likewise Yunker (1993) suggests that “in contrast to the EAR<sup>9</sup> Model it can be seen that the trajectory from initial mission statement through to formal NIAM Model can be **traced** at every point...This makes effective auditing of information models possible”<sup>10</sup> (p.16). Calway and Sykes (1995), identifying that NIAM offers no specific guidelines on the actual extraction of facts, investigated the possibility of further increasing the level of prescription by successfully applying discourse analysis to textual descriptions of the UoD. They concluded that it would be possible to document a method that would further assist analysts in undertaking the first step of the NIAM-CSDP.

However, the use of a prescriptive approach is not without its critics. Darke and Shanks (1995b) criticise the NIAM-CSDP on two grounds. Firstly, they remark on the limited sources of information that are used as the basis for requirements elicitation and secondly, on the nature of its first step. They argue that Step 1 “assumes that there is complete agreement as to the nature of the present situation, what problems exist, and what the information requirements of the new system are...There is, therefore, no consideration of how conflicting views could be resolved” (p.5). They continue “the set of elementary facts defined as the basis for the conceptual schema is considered to represent a neutral, objectively true description of a problem domain, expressed as a set of true propositions about reality” (*ibid.* p.5). However, NIAM proponents would not agree; van der Lek *et al.* (1992) for example, are quite clear that,

“NIAM’s central principle states that a database does not contain representations of objects from the real world but facts containing references (labels such as names, numbers, codes) to those objects (this is called a fact oriented approach). In other words; information modelling does not intend

---

<sup>9</sup>E(ntity) A(tribute) R(elationship)

<sup>10</sup> While auditability is no guarantee of success, the ability to trace a model’s development provides a means of highlighting erroneous assumptions and pinpointing significant decisions.

to model reality, but instead to model communication about reality. This principle is contrary to the object oriented view which states that a database constrains representations of real objects themselves” (p.2 (in translation)).

While it is correct that the NIAM-CSDP does not explicitly require the analyst to recognise conflicts or to view requirements definition as “a process of negotiation and interpretation” (Darke & Shanks, 1994b p.8), neither does it preclude it. Indeed, Darke and Shanks are commenting on the ‘*way of thinking*’ from which NIAM has sprung rather than a pre-defined restriction by which it is constrained.

Yunker (1993), a NIAM practitioner, for example, accepts that user requirements can be informal, incomplete, indeterminate, inconsistent and redundant. While Yunker does not provide any mechanisms for solving these problems, he clearly views NIAM as a useful and well-established procedure for formalising the information requirements once they have been agreed. Anecdotal evidence gathered from personal communication with NIAM-CSDP users<sup>11</sup> would suggest that they are well aware of the need for conflict resolution and negotiation. Like Yunker (1993), most of them viewed a prescriptive approach as allowing the construction of a formalised definition of requirements with no need of creative intervention by the analyst, once the set of requirements had been established as valid and appropriate for the current version of the UoD under consideration. Even Halpin (1995), while in no way accepting the subjectivist view, has to allow for semantic equivalence in conceptual schemas. He concedes that “given the informal nature of this initial step in modeling the UoD, it is not surprising that humans often come up with different ways of describing the same reality” (p.322).

Nevertheless it is clear that a prescriptive approach both stems from and encourages an objectivist view. It is not concerned with designing alternative solutions but on producing a single ‘correct’ solution. While this approach may be limiting (Darke and Shanks, 1995c), it does have some positive aspects which are discussed further in Chapter 7.

---

<sup>11</sup> The basis for this claim comes from personal conversations between the researcher and users of the NIAM-CSDP attending the NIAM-ISDM Conference in Albuquerque, New Mexico, USA in 1994.

### Differences in the way of working - classification

The second difference identified by Barden (1994) is that NIAM allows the analyst to record fact types without having to make a specific judgement on whether something is an entity, relationship or attribute. Indeed, the process of grouping or categorising data elements is one of the later stages of the modelling process. Modellers already familiar with the E-R approach will usually see an, apparently obvious, correspondence between the data elements in NIAM and the constructs of the E-R Model. Thus value type objects will often be construed as attributes, entity type objects as entities and predicates as relationships. However, despite the apparent similarities, within the NIAM-ISDM<sup>12</sup> these decisions are not finally made until the CSDP is complete and the model moves to the second stage of conceptual schema transformation.

Halpin (1995) discusses the transformation algorithm in detail but in essence the initial choice of constructs is revisited in the transformation stage and may well change. The decision to represent any data element as a particular construct is not ultimately the choice of the modeller but is the end result of an analytical process whereby the use of the element within the specific conceptual schema is determined. For example, an object *Department* with a reference mode of *DeptCode* may have been recorded as an entity type in the conceptual schema. However, if *Department* participates in only one role within the schema, perhaps as a property of the entity type object *Employee* then after transformation it will appear as an attribute of an *Employee* relation rather than as an independent relation in its own right.

Weber and Zhang's (1991) evaluation of NIAM identifies the 'semantic overload' that occurs through allowing the use of one construct to represent both *things* and *their properties*. However, they not only point out that the BWW model which they are using "does not indicate the predicted consequences of this" but also that, "NIAM advocates would argue the advantages...[as]...the overload allows users to avoid having to distinguish between entities and attributes which is a well-known dilemma in the ERM" (p.80).

---

<sup>12</sup>NIAM-I(nformation)S(ystems)D(esign)M(ethodology)

Most forms of the E-R 'way of working', force the modeller to make an almost immediate decision on the appropriate construct for a data element. Without making such a decision a modeller is unable to begin recording the information requirement. Once a decision has been made to include the concept *Department* within a model, the modeller is forced to decide whether it is to be recorded as an entity or an attribute. There are two problems with this early decision-making. Firstly while the modeller may change this decision at any time and for any reason, there is no specific procedure that ensures that the decision will be revisited. Secondly, for reasons outlined previously, the decision will often be influenced by inherently datalogical, probably relational, considerations.

As discussed above, Batra and Antony (1994) have observed that novice modellers are often anchored to their initial solutions and while best practice guidelines might advise them to question these decisions or to seek alternatives, there is no mechanism to encourage them to do so. This is less likely to be a problem with experienced modellers who have already induced their own rules for usefully classifying data elements although, here, problems of a different kind may emerge. In Chapter 5 it was observed that there is likely to be a bias towards equating useful entities with potential candidate relations and the re-use of patterns and previous solutions was noted in two studies (Batra & Davies, 1992; Simsion & Shanks, 1993) as being characteristic of 'expert' behaviour. It would therefore, seem very unlikely that, even an inexperienced, E-R modeller would view *Department* as anything other than an entity. Instructional material often includes examples of *Employee* and *Department* in which both are 'useful' entities, thus a re-useable pattern is learnt and *Department* would also fit the only definition of an entity that is usually offered, that an entity is something about which we wish to store information. In the general sense then, the decision would seem valid but in the specific UoD it may not be and an E-R modeller is unlikely to ever question this. In this way, a modeller is introducing datalogical, or more specifically relational, considerations from the very beginning of the modelling activity. If it was proven that information naturally possessed an inherently relational structure, or that users intuitively thought about their information in terms of relational structures, this might not be a problem and the search for the ideal tool for conceptual data modelling would not still be active.

So it would seem that the descriptive nature of E-R modelling ensures that there is an almost complete overlap between the analysis and design tasks of the conceptual

modelling activity as it becomes difficult to record the results of analysis without introducing a considerable element of design. It is difficult to see how such a model can be free of implementation considerations.

Yunker (1993) goes further and suggests that “the concepts<sup>13</sup> used for representation of a conceptual model fundamentally affect the nature of the procedures for requirements analysis”(p.1). He argues that the use of the constructs of entities, attributes and relationships preclude the possibility of creating a prescriptive method for the construction of E-R models. His reasoning is that “the attribute concept...is in fact determined by the combination of an information structure concept (which expresses an elementary fact proposition) as well as a constraint (which expresses an information base state or combination of permitted elementary facts)” (*ibid.* p.18). In other words to decide whether or not *Photocopier* is an attribute of *Department*, it is necessary to ascertain the constraints, e.g. whether a *Department* can have more than one *Photocopier*. However, in order to ascertain the constraints we need to have established by what form of constructs both *Photocopier* and *Department* are to be represented. In essence, in order to be able to identify the correct construct, the modeller needs to know the relevant valid constraints but in order to formulate the constraints, the construct must also be known. Thus, there is an inherent circularity in the dependencies that exist between certain constructs that can only be broken by taking an arbitrary decision and then testing out the hypotheses consequent to that decision.<sup>14</sup> This, Yunker (1993 p.19) maintains, explains why “no prescriptive procedures currently exist (or ever will) for the recording of initial information requirements in an EAR-based conceptual model”.

### **Differences in the way of communicating - natural language**

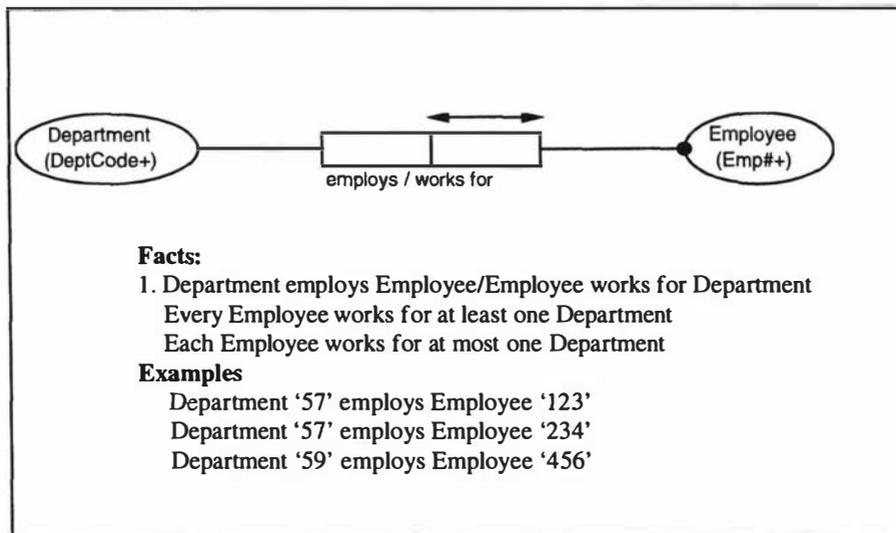
Barden (1994) also identified the use of a formalised subset of natural language, as another difference between NIAM and other methods. However, this is not strictly accurate. Several E-R/R methods, such as Information Engineering (Finkelstein, 1989) and the Oracle CASE\*Method standard (Barker, 1990) have mechanisms for presenting

---

<sup>13</sup> His meaning of the word ‘concept’ is synonymous with ‘construct’ as used in this study.

<sup>14</sup> The approach of synthesising 3NF relations from a ‘universal relation’ consisting of all required data elements could be seen as contradicting this view as it uses a highly prescriptive method. However, this approach is clearly specific to relational data structures and not considered a practical means of conceptual modelling (Kent, 1981).

model information as natural language, referred to as 'purpose descriptions' and 'Two Way Sentences' (TWS) respectively. What differentiates NIAM is partly the scale and the formality of the usage and partly the level of correspondence between the natural language sentences and the diagram. With the exception of a small number of constraints, which are most easily described graphically, all the information on the ORM diagram can be rendered into natural language form.

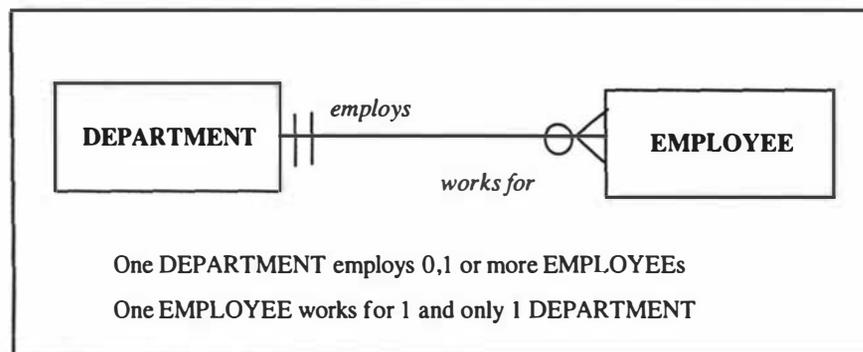


**Figure 10. Example of 'verbalization' report' from InfoModeler™**

Indeed if the steps of the CSDP are correctly followed the sentences do not need to be extracted from the model, they co-exist with it. Figure 10 provides a simple example of part of a report taken from InfoModeler™. A report such as this is automatically available from the tool at any point and is derived from the currently available information that has been input as a result of following the CSDP. A more comprehensive report can be produced including details of the data elements, such as data types, and more complex constraints, such as compound identifiers.

This co-existence is not a part of the E-R methods. The graphical representation, while supported by data dictionary information, can exist (and in some methodologies does exist) with no formalised natural language translation. Where relationships have been named in both directions, simple sentences can be derived as a process of interpretation of the graphical symbols and the element names. The two methods named above certainly encourage this and emphasise the benefits of doing so but many methods do not enforce it as an essential part of the construction process. The scope of the sentences is limited to

the associations between entities, and makes no reference to the identifying attributes, which often do not appear on the diagram either. Another disadvantage observed by Sharp (1993) is that, “these natural language business rules expressed as sentences must be generated and maintained separately ...the user is [thus] dependant on the modeler to keep the business rule list synchronized with the graphical EAR model” (p.3).



**Figure 11. Natural language interpretation of E-R/R constructs**

Figure 11 provides an example of a generic natural language interpretation of an E-R/R graphical representation. Initially there may appear to be little significant difference between the examples in Figure 10 and Figure 11 and certainly at this level of simplicity they would both seem to serve the same purpose satisfactorily. However, a complete description from the NIAM model would include the sentences for all data elements, which would include attributes and the associations in any generalisation/specialisation hierarchies thus providing a much richer description of the UoD. Additionally the NIAM description provides examples<sup>15</sup> with which to test out the hypotheses proposed by the model.

However, the most significant difference, already mentioned briefly, is not immediately obvious. The NIAM sentences, by their very nature, are an alternative view of the diagram and contain “all of the model content and approving one is equivalent to approving the other” (Sharp, 1993 p.6). On the other hand, the E-R sentences have to be extracted from the diagram by a process of interpretation. This extraction process is open to error and individual interpretation. As de Carteret and Vidgen (1995 p.373) observe,

<sup>15</sup> The use of examples is discussed later. See page 99

“the act of reading a data model is itself interpretative and subject to the same difficulties experienced when the modeller interpreted the situation to produce a data model”. Yet, this problem is rarely, if ever, addressed in texts and is not included as a necessary step in any common E-R approach. However, the process of interpretation is not clearly understood (Nordbotten & Crosby, 1996) and is too important to be left to intuition<sup>16</sup>. Siau *et al.* (1995,1996) for example, have hypothesised that expert modellers interpret a model by reading the syntactical constraints with little reference to the underlying semantics. They expect users, on the other hand, to interpret a model by referring to their semantic understanding regardless of the syntax. Some initial studies are now underway to test these theories.

Of course, NIAM’s use of sentences goes further than providing a natural language view of the schema. The identification of the elementary facts is firmly rooted in the language of the UoD, while the second axiom of the NIAM-ISDM is that all communication with the user is conducted in the users’ own language. Schouten (1994) considers that this ensures that “the persons involved know precisely what they are talking about” and that “the persons participating in the analysis, experience a sense of familiarity with the object of analysis throughout the whole process” (p.E2). This may be overstating the case but it seems reasonable to suggest that it is easier for users to actively participate in the conceptual modelling process if they are able to use natural language to do so (Leung & Nijssen, 1988; Sharp, 1994). It helps to minimise “users and managers assuming that the model as presented is correct because of their faith in the modeler and not because they can independently understand and verify the complete accuracy of the model” (Sharp, 1993 p.3). This, in turn, allows responsibility for the final approval of the conceptual model to rest with the users and not with the modellers. The use of natural language sentences also provides for long-term value, as they will always be accessible to future users, regardless of the presence of the original modeller or any graphical representations.

---

<sup>16</sup> A method to assist in the interpretation of E-R diagrams is described in Chapter 10.

### **Differences in the *way of communicating* - familiar examples**

The final difference identified by Barden (1994) that will be discussed here is the use of familiar examples from the UoD. Examples have three purposes in the NIAM-CSDP. Firstly they are used to formulate the initial example sentences, secondly they are used to specify the correct constraints on the elementary facts and thirdly they can be used to verify the final model and minimise ambiguity. There are obvious benefits of understandability in employing examples for verification, and of accuracy in using them to determine the constraints. However, it is their use within the initial sentences that is fundamentally different. NIAM begins with examples of real data formalised into standard sentence structures and then uses these structures to discern groupings and patterns within them. The examples are not collections of random data selected to test out certain hypotheses but rather they are at the very heart of the hypothesis creation. Indeed, without examples the NIAM-CSDP cannot begin. E-R methods, on the other hand, require the modeller to create patterns from generalised statements, isolated pockets of detail, acknowledged and unacknowledged assumptions and previous experience. Structures are then created from these imagined patterns, which may or may not be tested out against 'real' examples, depending on the experience, competency and confidence of the modeller.

### **An argument for integration?**

While superficial differences between the NIAM-CSDP and the E-R/R approach such as the different graphical representations are the most obvious, the comparisons discussed here, could be considered to be the deeper and more significant ones. They assist in a characterisation of the two approaches, which clearly suggests that they should be viewed as complementary techniques rather than alternatives.

The NIAM-CSDP has been shown to be significantly prescriptive in its approach, which may render it easier to learn and to follow even for those with little previous modelling experience or understanding of database design. This prescriptiveness also ensures a high degree of auditability, reproducibility and makes it possible to introduce quality assurance at each step. The use of natural language and real examples, as a fundamental basis of the method, allows for considerable user involvement and provides a natural

infological translation of the conceptual model that is straightforward to verify. Users can be largely isolated from datalogical considerations partly by removing the need for them to view the graphical representations and partly by allowing the modeller to postpone many design decisions until a relatively late stage in development. It can be criticised for being restrictive, non-creative, inflexible and working most effectively from an objectivist view of the world, which denies the possibility of truly alternative solutions.

The E-R approach exhibits very different characteristics. It is primarily a descriptive method that requires a long learning curve and a successful model will reflect not only the modeller's modelling expertise but also their understanding of implementational data structures. The method encourages production of a datalogical representation that can be difficult for users to recognise and is open to misinterpretation. Its strengths however, also lie in its descriptive nature. By relying on some element of trial and error it encourages the exploration of alternative and possibly innovative solutions.

Arguably, then, the NIAM-CSDP can be seen as a suitable tool for "the accurate recording of the user's information requirements [*which*] is or should be the main goal of conceptual modelling" (Yunker, 1993 p.2). It encourages modellers to behave as if there were an objective reality and to uncover and record existing information rather than to create new knowledge. Thus, despite the insistence of its name,<sup>17</sup> it would seem to exhibit properties more useful in an analysis rather than a design method. For example, as Simsion and Shanks (1993) observe, "there is no way of reaching the best design by proceeding mechanically from user requirements" (p.3), which of course is exactly what the NIAM-ISDM does in its later stages. It may well be that there are development situations where this non-creative, mechanical transformation may be appropriate but there will be others in which the results of analysis need to be subjected to a creative design process.

On the other hand, the E-R/R approach would seem much more suitable as a design method. In fact, it can be argued that it is inherently so, the initial activity of identifying entities being fundamentally one of design choice not of analytic discovery.

---

<sup>17</sup> i.e. NIAM-Conceptual Schema Design Procedure

Consequently, it may be most effectively utilised after the initial analysis of user requirements has been conducted.

## Summary

This comparison has attempted to show that if a clear distinction is made between the analysis and design stages of data modelling, then the two existing techniques of the NIAM-CSDP and the E-R/R approach, working together in an integrated way promises a tool significantly more useful than either of them in isolation. One alternative would be to enhance either method by the addition of techniques to fill the perceived gaps. However, it has been suggested that, "in general, any conceptual modelling language which contains information structures which dictate a grouping of elementary facts is not suitable for the capture of information requirements" (Yunker, 1993 p.21). Also, the fact that, despite various efforts (e.g. Eden, 1996), no prescriptive method for E-R has yet been accepted, suggests that it might represent something of a wild-goose chase. Conversely, the addition of descriptive design elements to NIAM would seem to be essentially the same as integrating the two approaches. An interesting precedent can be found in Kent's (1983) description of a proposed "fact-based data analysis and design" method which attempted to precede E-R design by an analysis of the facts in the UoD. He argues that by "focusing on the facts to be maintained in a database, we obtain a methodology...which is at once simpler and more powerful than other methodologies"(*ibid.* p.3). However, before a possible integration is outlined, one remaining thread requires investigation - the means whereby a conceptual data model may be evaluated.



# 8 Evaluating Data Models

*"We often fix on some set of [these] characteristics as "essential" to a model, with the rest being cosmetic variations that do not really matter. The trouble is, each of us is likely to fix on a slightly different set of essentials. Unless the underlying assumptions are very carefully exposed, many debates about these models are in danger of comparing apples and oranges." (Kent, 1978. p.99)*

## Introduction

Despite the considerable level of interest in the creation and use of conceptual models, there have been relatively few attempts (Krogstie *et al.*, 1995; Lindland *et al.*, 1994; Pohl, 1994) to define a systematic framework within which their quality can be evaluated. There have also been few attempts to build such a framework for conceptual data models (Kesh, 1995; Moody & Shanks, 1994; Shanks & Darke, 1996, Moody & Shanks, 1998). However, without such frameworks, it is difficult not only to compare and generalise the results of individual researchers but also to gain any holistic appreciation for the issues of quality as they relate to data modelling. Batra and Srinivasan (1992) have highlighted some of the issues surrounding this, commenting that "a number of studies do not mention the grading scheme and reliability of the scoring method" (p.412) which has been used to determine the outcome. They also comment that researchers have generally "defined and examined small pieces of the overall problem" (p.414). They omit to point out that not only do the studies often examine different characteristics of conceptual models but also, that even when the characteristics do overlap, there is no common evaluation approach. In addition, there has also been a tendency to equate model quality with syntactic correctness and thus much research misguidedly concentrates on the issue of 'building the product right' rather than 'building the right product' (Krogstie *et al.*, 1995).

## Previous studies

Most researchers appear to have adopted an essentially *ad hoc* approach to selecting evaluation criteria specific to their own particular experiment. Table 2 highlights the quality characteristics that the studies examined and the measurement instruments that were used to evaluate them.

Date	Authors	Tools	Quality	Measurement
1987	Shoval & Even-Chaime	Normalisation NIAM	1) best results, 2) ease of use 3) which preferred method	1) 'correct' solution 2) how long 3) subjective survey
1989	Jarvenpaa & Machesky	LDS, Relational	1) accuracy , 2) time 3) notational understanding 4) topdown/ bottom up	1) 'correct' solution 2) how long 3) comprehension questions 4) protocol analysis
1990	Batra, Hoffer & Bostrom	EE-R , Relational	1) accuracy in capturing semantics, 2) ease of use	1) 'correct' solution 2) subjective survey
1991	Batra & Davies	user choice leading to relational	1) process differences of expert v novices 2) correctness	1) protocol analysis 2) 'correct solution' (implicit)
1993	Marche	Relational	model stability	Changes in relations and attributes over versions
1993	Amer	E-R Relational	understandability by oversight users	discrepancy checking - no of mistakes made in identifying errors
1993	Bock & Ryan	EER OO	syntactic correctness	1) 'correct' solution
1993	Shanks <i>et al</i>	E-R	1) completeness 2) innovation 3) flexibility	1) support for specific functions 2) no of new nouns used as entities 3) level of generalisation
1993	Simsion & Shanks	E-R	1) completeness 2) innovation	1) support for specific functions 2) no of new nouns used as entities
1994	Batra & Antony	E-R	reasons for errors in modelling relationships	protocol analysis supported by use of 'correct' solution
1994	Shoval & Frummermann	OO and E-R	user comprehension	correct identification off true/false statements
1995	Hitchman	E-R	comprehension of semantic constructs in E-R	survey questionnaire 'correct' solution (implied)
1995	Kim & March	EE-R ORM	1) correct 2) consistent 3) complete 4) comprehensible	1, 2 and 3) 'correct' solution 4a) no of correct answers 4b) no and type of identified errors
1997	Shanks	E-R	1) correctness, 2) completeness, 3) innovation 4) flexibility 5) understandability 6) overall quality.	1) correct syntax 2) 'correct' solution 3) new nouns as entities 4) subjective assessment 5) subjective 6) subjective
1997	Shoval	EER OO	1) understandability 2) correctness of schemas 3) time taken to complete 4) designers preference	1) No True/False statements answered correctly 2) 'correct solution' As per Batra <i>et al</i> 1990 3) Time recorded 4) Designers subjective assessment on 7 point scale

**Table 2 Evaluation criteria used in conceptual modelling studies.**

It would seem from this table that there is one common element, the use of a 'correct' solution, among the measurement instruments. All but two of the studies for which it was appropriate (Shanks *et al.*, 1993;Simsion & Shanks, 1993), used a grading scheme based on a benchmark 'correct' solution to test for such elements as "correctness" (Batra

& Davies, 1992), “completeness” (Shanks, 1997), “consistency” (Kim & March, 1995) and “best results” (Shoval & Even-Chaime, 1987). A more detailed discussion of these characteristics is postponed until later but it is important to recognise that these terms sometimes indicate semantic completeness, sometimes syntactic correctness and at other times no distinction is made between them.

All but one of the studies used different scenarios or case studies<sup>1</sup>, nine of which accompanied the published research, at least in part, (Batra & Antony, 1994; Batra & Davies, 1992; Batra *et al.*, 1990; Bock & Ryan, 1993; Jarvenpaa & Machesky, 1989; Kim & March, 1995; Shanks *et al.*, 1993; Shoval & Frumermann, 1994; Simsion & Shanks, 1993; Shoval, 1997). However, only two included the ‘correct’ solutions in their entirety (Batra *et al.*, 1990; Bock & Ryan, 1993), and one in part (Batra & Antony, 1994). In addition, only three reproduced the grading schemes that had been used to score errors (Batra *et al.*, 1990; Bock & Ryan, 1993; Kim & March, 1995). It is thus difficult to judge how similar the use of a ‘correct’ solution really is. The level of complexity of the task, the grading scheme used, the strictness with which the grading scheme is applied, the inherent difficulty in creating a ‘correct’ solution and the issue of semantic equivalence, are all factors which suggest that it is unlikely that the measurement was applied in a consistent manner. Some of the issues pertinent to this are discussed below.

### **Measuring correctness**

Firstly, there is no consistency between the complexity of the scenarios that participants are asked to model. Shoval and Even-Chaime (1987) used four scenarios, two of which they described as ‘simple’, involving a “relatively simple DFD, with a single data store and a few dataflows and examples only” (p.36). The other two scenarios are “more complicated” involving a “more detailed DFD which included two data stores, more data flows and examples, more data elements and more complex relationships/dependencies between them” (*ibid.* p.36). Despite this information, Batra and Srinivasan (1992) comment that there is no means of judging the relative

---

<sup>1</sup> Bock and Ryan (1993) used the same scenario as Batra *et al.*, (1990)

complexity of the scenarios<sup>2</sup>, nor of comparing them with other studies. Jarvenpaa and Machesky (1989), report expecting 5 entities, 13 attributes and 8 relationships while Batra *et al.* (1990) required 9 entities, 2 sub-entities, 23 attributes and 8 relationships in their relational solution. Batra and Antony (1994) have only 4 entities and 3 relationships in the solution that they provide while Simsion and Shanks (1993) report an average of 11 entities in their study with the range extending from 5 to 31. Hitchman (1995), on the other hand, points out that his study used scenarios that were “designed to be simple descriptions of recursion, entity sub-types, orthogonal entity sub-types and exclusivity” (p.35). He also comments that in grading them “the interpretation of ‘correct’ was strict as “modellers could be expected to be precise in simple scenarios” (*ibid.* p.37). There is no indication of how rigorous the interpretation of ‘correct’ was in the other studies and only two specifically address the issue of consistency between the graders and give inter-rater reliability scores<sup>3</sup> (Batra *et al.*, 1990; Bock & Ryan, 1993). Clearly there is no consistency in the level of complexity of the tasks that were undertaken by the participants and for most of the studies there is no indication of how correctness was scored or what degree of correctness was deemed acceptable.

Secondly, a number of the researchers themselves voice some general concern over this means of judging quality. Shoval and Even-Chaime (1987), describing their use of what they describe as the ‘gold standard’ approach, comment that with “...all the reservations that can be made, we judge it as the most acceptable surrogate measure for quality that could be applied in the experimental environment”(p.37). Other authors appear to agree with the view that the “idea of a unique correct solution is generally central to domains such as physics, (and) linear programming” but for conceptual modelling “it is more realistic to think...in terms of degree of correctness” (Batra & Davies, 1992). Batra *et al.* (1990 p.130) go further in explicitly stating that “there was no restriction on the number of correct solutions if they translated to the same semantics.” Unfortunately they do not specify how they determined semantic equivalence. Likewise, Batra and Antony (1994) are clear that “alternative solutions [*to the ‘correct’ one*] were permitted

---

<sup>2</sup> The authors themselves remark that, “it turned out that of the two complex tasks, one was considerably more complex, as it took much more time to do” (Shoval & Even-Chaime, 1987)

<sup>3</sup> Shanks (1997) also provides this but does not use a ‘gold standard’ approach to assessing correctness.

so long as they captured the semantics” (p.61). In the light of Simsion’s observation that there is no formula for creating “the ‘best’ classification scheme or even for recognising it when we do” (Simsion, 1994 p.124) and Moody and Shanks’ (1994 p.7) statement that it “is impossible to say in any absolute sense that one data model is better than another, irrespective of context”, it seems that the ‘gold standard’ method of judging correctness may be less than satisfactory. This would seem particularly so when it is the primary means of measurement and no indication is given of how semantic equivalence has been determined or graded.

### **Alternative Approaches**

An alternative approach to measuring semantic correctness or completeness is detailed in two papers (Shanks *et al.*, 1993; Simsion & Shanks, 1993). In the first, the authors focus on conceptual data modelling as primarily a design activity<sup>4</sup> and state clearly that there can be no ‘correct’ answer against which the data models can be evaluated. They specifically reject the idea of the ‘gold standard’ solution as being inappropriate given the notion of alternative solutions not only being possible but also highly desirable. Having also defined completeness as the degree to which the model contains the information needed to support the functionality of the required system, they propose and use a measurement instrument based on how effectively each data model can support four key business functions. This method is also used in the subsequent study (Simsion & Shanks, 1993) where each function is rated as zero, one or two reflecting “no support, partial support or complete support” (p.7) for the key functions. Although no specific problems are raised by the authors, they do note that it “was a time consuming process, impeded by the lack of entity descriptions”( *ibid.* p.7) even though it was undertaken by someone with “over ten years consulting experience as a specialist data modeller” (*ibid.* p.7). It is interesting that in a more recent study, Shanks (1997) has abandoned this method in favour of the ‘correct solution’ approach.

---

<sup>4</sup> See the discussion in Chapter 4.

## Evaluating comparative studies

Another frequently found common element in the reported studies<sup>5</sup> is the presence of the Relational Model as one of the Data Models under consideration and this leads Batra and Srinivasan (1992) to remark that,

“Representations that are prepared using semantic data models may have to be converted to ones using the relational data model. If a semantic data model is found to lead to better user performance, there is still the empirical question of whether the quality of data representation obtained after the designer converts it to the relational form is better than if the designer were to directly prepare a relational representation” (p.412).

This concern, albeit in a different form, would also seem relevant to studies comparing different semantic Models (Bock & Ryan, 1993; Kim & March, 1995; Shoval & Even-Chaime, 1987; Shoval & Frumermann, 1994), or to those where the choice of representation is left to the modeller’s discretion (e.g. Hitchman, 1995; Shanks *et al.*, 1993; Simsion & Shanks, 1993). While a satisfactory solution may in part depend on whether the study has a datalogical or infological emphasis, it is clear that some means of comparison must be established. The tasks set by Shoval and Even-Chaime (1987) required participants to create either a set of “normalised record types”, for the normalisation solutions or a set of the record types that resulted from applying a grouping algorithm, for the Information Analysis solutions. While it is clear that all solutions were graded against a ‘correct’ solution, it is not stated whether the same solution was used for the different formalisms although this seems likely. Kim and March (1995) on the other hand appear to have used the ‘gold standard’ approach independently for each formalism and then compared the resulting scores. Hitchman (1995) adopts yet another approach and reports that the “diagrams for the scenarios were coded according to the author’s understanding of the model presented by the respondent. The model was ‘registered’ if the survey analyst could clearly recognise that the semantic constructs in the diagram modelled the given scenario” (p.35).

---

<sup>5</sup> See the previous chapter for a detailed discussion of the comparative studies from which these observations have been made.

### **Desirable characteristics of a conceptual data model**

Correctness, whether judged to be syntactic correctness, semantic completeness (Shanks, 1997) or both, was by far the most common characteristic measured in these studies. Even where it was not a major focus, it was understandably considered an essential pre-requisite to measuring other factors. There is, after all, little point in concluding that a modelling formalism is easy to use, for example, if the 'quality' of the outputs is very low. However, it is not usually clear in the literature, whether the responses from a participant whose model showed little quality was still recorded as valid and neither do the studies generally report any correlation between these kinds of factors. Understanding, in some form, either in terms of semantics i.e. 'reading' a model (Amer, 1993; Kim & March, 1995; Shoval & Frumermann, 1994) or syntax, i.e. using appropriate notation, also featured quite prominently. Other characteristics are shown in Table 2 from which it can be seen that Shanks' (1997) study alone attempted to create a comprehensive list of characteristics and to measure each of them, including "the 'transcendental' properties of the data models which are not included in the other quality factors" (p.68).

It would seem then that there are three issues that have not been satisfactorily addressed in evaluating the quality of data models. There appears to be no consensus on what desirable characteristics are required of a conceptual data model, no agreement on how those characteristics can be measured and no direct method of comparing the quality of a data model constructed with one formalism with one constructed using another. The situation is compounded, however, by yet another missing element.

### **Conceptual Data Model - definitions**

In order to create a meaningful set of evaluation criteria it is essential to determine the purpose, or purposes, the conceptual model is intended to serve. As noted earlier there is a singular lack of consensus on these issues and nowhere is this more clearly demonstrated than in the studies that purport to examine it. All but two of the studies in Table 2 ostensibly deal with the 'conceptual model'. However, their definitions, sometimes only implicit in the general description of their research area, range from the modelling of entities, attributes and relationships (Batra & Antony 1994) to a non-

implementation representation of user information (Shanks *et al.*, 1993) and a formal representation of the “data structure of reality” (Shoval & Frumermann, 1994 p.28). While the differences may seem subtle, their impact of the choice of measurement instruments and on the interpretation placed on any results can be significant. Table 3, provides the definitions determined from each study and shows whether or not these definitions were explicitly described.

Date	Authors	Stated Focus	Definition of conceptual model	Explicit
1987	Shoval & Even-Chaime	database design	N/A	
1989	Jarvenpaa & Machesky	logical data model	model must conform to user view with no consideration of computer storage	Yes
1990	Batra, Hoffer & Bostrom	conceptual data model	abstraction of the real world data pertinent to an enterprise	Yes
1991	Batra & Davies	conceptual data model	capture the data structure and semantic constraints with no implementation details. Use for communication	Yes
1993	Marche	logical relational schema	N/A	
1993	Amer	conceptual database model	representation of organisation's database to help humans to understand contents of the database	Yes
1993	Bock & Ryan	conceptual data model	representation of data and data relationships in an implementation dependent manner	Yes
1993	Shanks <i>et al</i>	conceptual schema	non-implementation representation of user information “describe problem domain and propose classification schemes for it.	No
1993	Simsion & Shanks	conceptual schema	a means of describing data to support a particular business area	Yes
1994	Shoval & Frumermann	conceptual schema	formal means of representing the data structures of reality.	Yes
1994	Batra & Antony	conceptual database design	modelling of entities, attributes and relationships - no strict differentiation between it and logical design as both need to capture semantics of the application	Yes
1995	Hitchman	data modelling by practitioners	implied as a part of the database design process	No
1995	Kim & March	conceptual model	formal representation of the enterprise reality. Serves as a communication tool.	Yes
1997	Shanks	conceptual data model	precise, unambiguous representation of organisational information requirements	Yes

**Table 3 Definitions of ‘conceptual model ‘ in comparative studies**

Implicit, in almost all of these definitions, is the division between the datalogical and infological purposes of a conceptual data model but, with the exception of Kim and March (1995), a clear distinction between them in terms of evaluation, is rarely made. The various quality frameworks that are discussed below undoubtedly do recognise the importance of the infological properties of a conceptual model and consequently are more likely to provide a well-balanced set of criteria.

## Conceptual Model Frameworks

### *Lindland et al. (1994)*

Until the publication of Lindland *et al.*'s paper in 1994, research into determining quality in conceptual models had focused almost exclusively on the quality of the end product and resulted in a number of lists of desirable features and properties. Shanks and Darke (1996) compiled a table of research in this area, a version of which is shown at Table 4. Lindland *et al.* (1994) observe that previous lists of desirable properties for conceptual models have not provided a “systematic structure for evaluating them” and consequently propose a framework that not only “identifies major quality goals but gives the means for achieving them” (p.43).

<i>Date</i>	<i>Approach</i>	<i>Purpose</i>	<i>Features</i>	<i>Type</i>	<i>Focus</i>
1985	Roman	Defining properties of requirement specifications	Properties linked to their use in the design process	List	Theory
1992	Batini <i>et al</i>	Improving the quality of a database schema	Quality features of a good schema, schema transformations	List	Theory
1994	Lindland <i>et al</i>	Understanding quality in conceptual modelling	Linguistic base, separation goals from means	Framework	Theory
1994	Moody & Shanks	Evaluating the quality of E-R models	Quality Factors, strategies and evaluation methods	Framework	Practice
1994	Pohl	Defining goals and process dimensions for requirements modelling	Specification, representation and agreement dimensions	Framework	Theory
1994	Simsion	Defining quality features in E-R models	Design and evaluation of alternative models	List	Practice
1995	Krogstie <i>et al.</i>	Understanding quality in conceptual modelling	Extends Lindland <i>et al.</i> Using agreement goal and social construction theory	Framework	Theory
1996	Shanks & Darke	Bring together theoretical and pragmatic quality factors	Amalgamates frameworks of Krogstie <i>et al</i> and Moody & Shanks	Framework	Theory and Practice

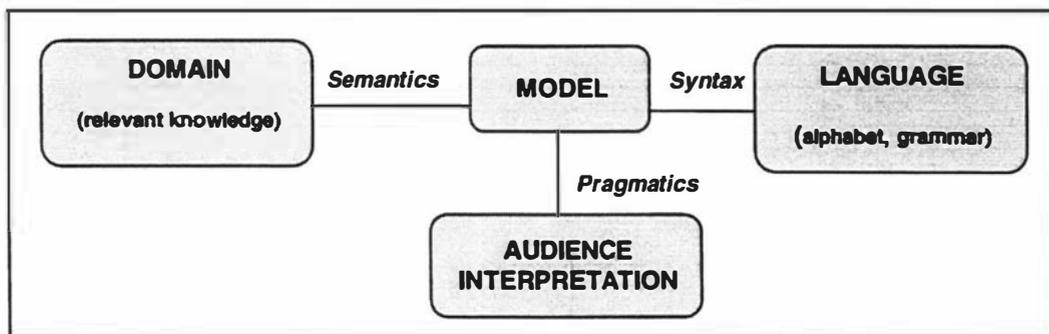
**Table 4 Approaches to quality in conceptual modelling adapted from Shanks and Darke (1996)**

Lindland *et al.*'s (1994) proposed framework is primarily based on the linguistic concepts of syntax, semantics and pragmatics, in recognition that “modelling is essentially making statements in some language” and it is thus able to subsume all the characteristics they had listed from previous studies. In common with some of the discussion above they also, “observed several significant trends:

- Many definitions are vague, complicated, or in some cases, even lacking...
- The list [*of characteristics*] is unstructured and the properties are partly overlapping...

- Specification properties are mixed with language and method properties...
- Some properties presuppose the existence of a design and even an implementation...
- Some goals are unrealistic, even impossible to reach” (*ibid.* p.43).

Their framework is intended to deal with those issues as well as making a clear distinction between the goals and the means, i.e. “by separating *what* you are trying to achieve...from *how* to achieve it” (*ibid.* p.42).



**Figure 12. Lindland *et al.*'s (1994) Framework**

Each aspect of their framework, illustrated in Figure 12, is clearly defined and sufficiently generalised to be applicable to any conceptual modelling activity not just those related to data. **Language** is the statements that can be made according to the specific syntax and is constructed using an alphabet (the set of modelling constructs) and a grammar (the rules that govern the use of the constructs). In addition, the language also has semantics, which define the meaning of the constructs and thus enable meaning to be derived from them. The definition of **Domain** is closer to the relational notion of a pool of allowable values than to its more generalised use as a description of an application area. Thus the framework's domain consists of all the possible statements that would be correct and relevant for solving the problem. The set of statements actually made in the language or which can be derived from such statements is the **Model**. The audience is defined as all those who need to understand the model and **Audience Interpretation** is then the set of statements that the audience thinks the model contains. The connections between these four “cornerstones” provide the specific linguistic links that create the framework. The **syntax** link relates the model to the modelling language by describing relations among language constructs without considering their meaning. The **semantics** link relates the model to the domain by

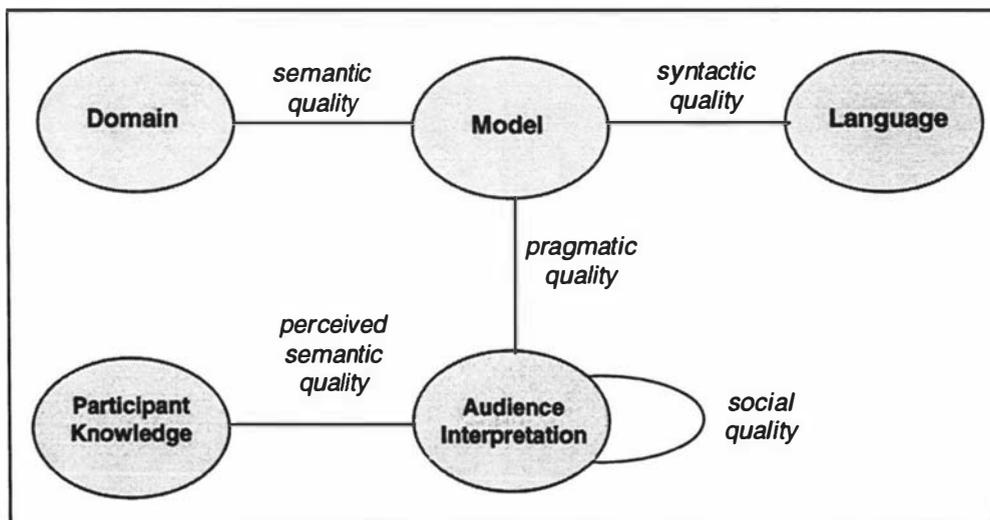
considering not only syntax but also relations among statements and their meaning. The **pragmatics** link relates the model to the audience by considering not only the syntax and semantics but also how the audience will interpret them,

Having constructed this framework and given these definitions Lindland *et al.* (1994) are able to specify a structured set of goals for the three linguistic dimensions that they have identified. *Syntactic quality* has only one goal, that of syntactic correctness. In other words, all statements made within the model are made according to the rules of the syntax. The appropriate activity for measuring this quality is syntax checking, made possible by the existence of formal syntax. *Semantic quality* on the other hand has two goals, validity, whereby all statements contained in the model are correct and appropriate to the domain and completeness, whereby “the model contains all the statements about the domain that are correct and relevant” (*ibid.* p.46). Three proposed activities for achieving these goals are consistency checking, statement insertion and deletion. *Comprehension* is the one pragmatic goal, meaning that the audience, or at least the relevant section of the audience, has understood the model, either in whole or part, as appropriate. Comprehension is thus further defined as the situation where the relevant part of the model has been understood by the appropriate sub-set of the audience. These latter two qualities are further qualified by the notion of feasibility in recognition of the fact that creating a ‘perfect model’, even if it were possible to do so, would require an unacceptable amount of resource. A trade-off is thus introduced “between the benefits and drawbacks for achieving a given model quality” (*ibid.* p.45) and is allowed for in the equations proposed for measuring the various qualities.

One final contribution that the authors make to this debate is the noting of three other factors, which, although not measurable within their framework, may well affect the quality of the model that is produced. These are the notions of appropriateness: between the language and the domain, the language and the audience and the audience and the domain. The extent to which the language itself is able to satisfactorily describe the domain and the level of familiarity of the audience both with the domain and with the language undoubtedly affects the overall quality of the modeling process. As they observe their framework does not deal explicitly with these notions but assumes “that an acceptable level of appropriateness has been achieved” (*ibid.* p.46).

**Krogstie et al (1995)**

Krogstie *et al.*'s (1995) framework extends that of Lindland *et al.* (1994) by incorporating one further concept, *participant knowledge* of the domain, and two further dimensions, *perceived semantics* and *social agreement* as illustrated in Figure 13. Shanks and Darke (1996) explain this last dimension as being derived from "social construction theory in which reality is considered to be subjectively constructed via our beliefs, values and perceptions. Thus each individual social actor perceives the world in a way specific to him or her creating their own "local reality". Organisations consist of social actors sharing a common pool of values, an inter-subjective reality" (p.5). Participant knowledge then is the collection of all sub-sets of statements where each sub-set contains all possible correct and relevant statements for a particular actor or group of actors.



**Figure 13. Concepts in the framework of Krogstie *et al.* (1995)**

*Semantic quality* is seen by Krogstie *et al.* (1995) as impossible to establish or check directly as it is an ideal solution that is inaccessible to the participants. *Perceived semantic quality* is considered more relevant and is defined as that which is actually observed and compares the participants' knowledge of the domain with their interpretation of the model. It thus seeks to incorporate the audience-domain appropriateness, identified by Lindland *et al.* (1994) as external to their framework. The goals are *perceived validity* and *perceived completeness* while the activities for achieving them are enabled not only by the formal semantics and modifiability of the model but also by audience training in knowledge both about the domain and the model.

*Social quality* has the goal of feasible agreement between the actors, where inconsistencies in the various actors' interpretations of the model are resolved. *Relative agreement*, where audience interpretations may differ but remain consistent is seen as more realistic than *absolute agreement* whereby all interpretations must be the same.

As Shanks and Darke (1996 p.5) observe this framework "organises quality in conceptual modelling into components relating to the model as a linguistic artefact...and components relating to the quality goals and means of achieving them." In addition it recognises that particular participants see particular quality goals as relevant. Shanks and Darke (1996 p.5) conclude that the framework "provides a sound theoretical base for understanding the notion of quality in conceptual modelling." Kroenke (1992) advises novice modellers to "learn to ask these questions: Does this model accurately reflect the users' perceptions and mental models of their world? Will it help users respond consistently and successfully with one another and with their clients?" (p.118). It would certainly seem that this framework could provide a structure within which these questions might be answered.

While clearly providing a comprehensive theoretical base, Krogstie *et al.*'s (1995) framework is deliberating uninstantiated and is not intended to provide a formal method for evaluating any particular type of modelling tool or process. Moody and Shanks (1994), on the other hand, developed a comprehensive framework specifically to evaluate conceptual E-R, or more correctly E-R/Relational hybrid models and were clear as to the need to develop a framework that was not only rigorous but also practical and useable. Their framework was also targeted directly at the practice of conceptual modelling, and was intended to provide "practitioners with a coherent approach to evaluating the quality of data models and choosing between alternatives in place of the *ad hoc* methods they currently use" (*ibid.* p.3).

#### ***Moody and Shanks (1994)***

Moody and Shanks<sup>6</sup> (1994) begin by defining quality in line with the IEEE standard as the totality of features and characteristics of a product or service that bears on its ability

---

<sup>6</sup> This paper has been refined and published as Moody and Shanks (1998). However, the substantive content, particularly as it relates to this discussion, is largely unchanged.

to satisfy given needs. With this in mind, they include four elements in their framework: **qualities** seen as the desirable properties of a model, **metrics** to provide a means of measuring each quality, **weightings** to determine the relevant importance of any given quality to any specific context and **strategies** to improve the value of any given quality. Unlike the previous frameworks, Moody and Shanks (1994) then proceed to populate their framework by identifying six qualities together with appropriate metrics<sup>7</sup>.

The first is **simplicity**, taken as a measure of the number of constructs required to express the required semantics. It defines the appropriate metric as being the sum of the number of entities in a model and the number of relationships with the objective of achieving a minimum value. Simplicity is considered a valuable characteristic on the basis that simpler models are generally more flexible, easier to implement and easier to understand. **Completeness**, in the sense of semantic correctness is the second quality and there are four proposed metrics; subjective ratings from users and industry specialists, the mapping of the required process to the data model and finally by mapping the model to an existing software package that reflects the user requirements. The third quality is **flexibility**, i.e. the ease with which the data model can be adapted to changes in requirements. All the proposed metrics for this quality rely on subjective assessment by experts, specifically, senior management, industry experts and expert data modellers. **Integration** relates to how well the data model fits with other organisational data models, such as a corporate data model. Suggested metrics include, comparing the new model with existing ones to identify duplicate or overlapping entities, structural conflicts and domain conflicts. **Understandability** and **implementability** complete the list with the first being measured by the ease with which the users of the model understand its concepts and structures. The latter specifically relating to the creation of an effective database is measured in terms of the risks and costs of building it.

### ***Shanks and Darke (1996)***

Moody and Shanks (1994) framework is an important milestone in that practitioners are offered, for the first time a means of resolving two key problems, the need to choose

---

<sup>7</sup> Weightings and strategies are not instantiated.

between a number of alternative models and the need to understand and accommodate the different views of the various stakeholders. Shanks and Darke (1996), recognised that, while this framework provided “the components to support the evaluation of models in practice”, it also lacked a “sound basis in theory” (p.1), and so have proposed a composite framework. This draws on the theoretical strengths of Krogstie *et al.* (1995) while also incorporating some of the practical emphasis found in Moody and Shanks (1994). In defining this composite Shanks and Darke are addressing the need highlighted by Batra and Marakas (1995) for a greater synergy between theory and practice and the need for techniques based on sound theoretical principles but which are still pragmatic and useable.

<i>Date</i>	<i>Author</i>	<i>Quality Goals</i>	<i>Metrics</i>
1994	Lindland <i>et al.</i>	1) Syntactic - correctness 2) Semantic - feasible validity 3) Semantic - feasible completeness 4) Pragmatic - comprehension	1) Syntax check 2) All statements correct and relevant for domain 3) All correct and relevant statements are included 4) Each audience understands relevant sub-set of model
1994	Moody & Shanks	1) Simplicity 2) Completeness 3) Flexibility 4) Integration 5) Understandability 6) Implementability	1) Entities + Relationships 2) Subjective ratings and mapping to processes 3) Subjective ratings 4) Subjective ratings and no of conflicts with other models 5) Subjective ratings 6) Analysis of costs and risks
1995	Kesh	1) Suitability 2) Soundness 3) Consistency 4) Conciseness 5) Completeness 6) Cohesiveness 7) Validity 8) Usability - user 9) Usability - designer 10) Maintainability 11) Accuracy 12) Performance -efficiency	1) Subjective ratings(a) 2) Subjective ratings (b) 3) Deducting points for each inconsistency (c) 4) No of relationships 'additional' to minimum required.(d) 5) Data required by queries/reports not in model (e) 6) Size of primary identifier (i.e. Primary Key) (f) 7) No of required attributes placed in 'right' entities (g) 8) = (a+b+d+e)/4 9) = (b+c+e+f+g)/5 10) = (b+d+f)/3 11) = (c+e)/2 12) = (d+e)/2
1995	Krogstie <i>et al.</i>	1) Syntactic - correctness 2) Semantic -perceived validity 3) Semantic - perceived completeness 4) Pragmatic -feasible comprehension 5) Social - feasible agreement	1) Syntax check 2) All statements correct and relevant for perceived domain 3) All correct and relevant statements are included 4) Each audience understands relevant sub-set of model 5) No of conflicts between different actors' interpretations
1996	Shanks & Darke	1) Correctness 2) Perceived validity 3) Perceived completeness -completeness 4) Comprehension -simplicity, understandability 5) Feasible agreement - integration 6) Flexibility	Nothing further identified Assumed to be as detailed for individual frameworks.
1998	Moody & Shanks	1) Simplicity 2) Completeness 3) Flexibility 4) Integration 5) Understandability 6) Implementability 7) Correctness	1) Entities + Relationships 2) User reviews, process mapping, scenario analysis 3) Subjective ratings by various reviewers 4) Comparison with other models, business area reviews 5) User and developer reviews, scenario analysis 6) Analysis of costs and risks, developer review 7) Check of syntax

**Table 5 Goals and Metrics of proposed quality frameworks**

Shanks and Darke (1996) identify a number of overlapping concepts between the two frameworks and map them as shown in Table 5. However, they argue that although the Moody and Shanks' (1994) quality factors could be subsumed by the generic goals or properties of Krogstie *et al.* (1995), it is more beneficial to recognise them and their mappings separately. In consequence the meta-model of the framework that they include clearly shows that theoretical goals and properties map directly to practice-based quality factors, which, in line with Moody and Shanks' (1994) view, may be assigned a weighting factor relative to the model's context.

Shanks and Darke (1996) also report on the use of a hypertext tool, the Data Model Quality Advisor, to facilitate the evaluation and comparison of up to three alternative models. This tool, which provides an explanation facility for all aspects of their composite framework, also allows for the allocation of weightings to quality factors. While not discussed in any detail, they report on the results of an empirical study to examine the usability and usefulness of their framework. These seem to suggest that while the "theory-based components were seen to be of less direct relevance to practitioners", nevertheless, completeness, understandability and correctness were rated by the participants as the three most important quality factors which, of course, "correspond to the three goals of the theory-based framework components" (p.11).

### ***Kesh (1995)***

The framework provided by Kesh (1995) is comprehensive in that it seeks to measure aspects of model structure, content and behaviour (e.g. usability and maintainability) and his paper includes a worked example of a simple data model showing how the calculated scores can highlight potentially problematic areas. While the twelve characteristics, which he describes, are broadly similar to those in the other frameworks his choices of measurement instruments are open to debate and refinement. The underlying, and stated, assumption that all relationships are binary leads to a less than satisfactory data representation in which a fairly common ternary relationship construct between three entities appears as three binary relationships. The metrics relating to conciseness ignore the situation where two or more relationships may be required between two entities by insisting that the minimum number of relationships required by a model will always be  $n-1$ . In addition, those relating to cohesiveness when applied to the example problem merely result in the creation of a surrogate key without addressing

the lack of referential integrity inherent in the constructed model. However, the completeness with which this framework is described provides a significant and useful contribution to the evaluation debate.

### Testing the Frameworks

The generic frameworks of Lindland *et al* (1994) and Krogstie *et al* (1995) have not been instantiated. The other three (Moody & Shanks, 1994; Shanks & Darke, 1996; Kesh, 1995) do not appear to have been tested empirically, although Kesh (1995) indicates that such a study is under way. However, Shanks (1997) does report a study in which a number of different quality factors were used to evaluate conceptual models. The study was designed to study the differences in the quality of conceptual models created by novice and expert practitioners and utilised six quality factors scored with a combination of objective measures and subjective ratings. Utilising a combination of the goals discussed above, Shanks identified the following components: -

- **Correctness** of syntactic representation, i.e. the degree to which the data model conformed to the syntax rules of the particular notation used (Batra *et al.*, 1990; Kim & March, 1995). This was evaluated as the number of major and minor errors found in the models. The overall correctness was obtained by averaging the participant's performance for each modelling construct.
- **Completeness** of the perceived semantic content (Krogstie *et al.*, 1995), i.e. the extent to which the data model supported user requirements. The evaluation instrument was based on comparison of the model to a 'correct' solution developed by the author with semantically equivalent constructs allowed. The overall completeness is obtained by averaging the participants' performances for each modelling construct.
- **Innovation** was "the extent to which 'new concepts' which are relevant and valid are introduced into the model" and is based on Shanks *et al.* (1993). It was measured by developing a list of nouns from the case study narrative and then counting those entity types in a model that did not correspond to any of the nouns on the list. The ratio of 'innovative' entity types to the total number of entity types in each data model was used as a measure of the level of innovation.

- **Flexibility** was defined as the ease with which the data model can reflect changes in requirement without changing the model itself. This was measured by experts using a seven point likert scale.
- **Understandability** or the ease with which concepts and structures of the model can be understood was also measured using a seven point likert scale.
- **Overall Quality** was another subjective rating applied to the whole model. It was included as an attempt to account for those elements of a model that may not be included under any of the specific quality factors.

An interesting result of the study was the low level of measured understandability. Although confirming previous studies which had suggested that conceptual models were not as easy to use as had been previously thought (Goldstein & Storey, 1990; Hitchman, 1995), the measured level of understandability was very low, “below the mid-point of the likert scale” for both novices and experts. There was also a poor consensus between the reviewers on this measurement and not surprisingly a strong positive correlation between understandability and correctness and a strong negative correlation between understandability and innovation. While noteworthy in themselves these findings also have implications for the evaluation process itself. If the models are demonstrably difficult to understand, how solid are any measurements that rely on human interpretation of them?

Moody and Shanks (1998) comment that their framework, initially set out in Moody and Shanks (1994) has been used “both to evaluate individual models as part of application development projects and to implement data model quality assurance procedures in organisations” (Moody & Shanks, 1998 p108). They do not provide details of these evaluations although they do comment on a number of benefits the use of the framework has brought. These include encouraging the development of alternative models, the ability to determine productive areas for improvement and the means to make effective choices between alternative structures.

## Summary

The current literature contains a rich resource of comparisons. Some studies have sought to compare the E-R or EER Model with other conceptual Models or with one of the classical Models, usually the relational. Some have focussed on the builder's perspective, attempting to suggest which Model provides richer or more precise schemata, others on the learner's perspective to discover whether any one Model is superior in terms of ease of use or learning. A few have concentrated on the user's perspective, to ascertain which of the Models leads to best comprehension of a model's contents. The results of these studies cannot be easily amalgamated because of their differing research methodologies and evaluation criteria. Not surprisingly then the results as Shoval and Frummermann (1994) remark, are "are not always clear-cut or consistent, (although) there is a tendency to agree that EER is superior to other, record-based and conceptual models"(p.29).

Finally, there are some interesting issues that arise from the body of research discussed in this chapter. Almost all the comparative studies which deal with the E-R Model use variations on the Chen notation despite the fact that this is poorly supported in current CASE tools and not widely used outside of academia. Secondly, those studies that evaluate the Relational Model use either a tabular or textual representation despite the fact that it is very common practice in industry to use a simplified version of the Chen notation to provide a graphical view of such a model. Consequently, much of the research completely ignores the E-R/Relational hybrid which most practitioners use as a matter of course to represent their conceptual models, which being normalised, with primary keys and resolved many to many relationships, are often indistinguishable, except perhaps in detail, from their logical (relational) models. It also ignores the growing trend in database texts to teach students this hybrid, sometimes exclusively. While the research may then provide some interesting, if idiosyncratic, results it is unlikely to be of very much use to those interested in providing useful conclusions to the Information Systems industry. Most importantly there do not appear to be any attempts to match the differing formalisms to appropriate activities and to integrate them into a more effective overall framework. Such an attempt is the subject of the remainder of this study.



# 9 INTECoM: an integrated framework

*“Thus the truth of things may be this: useful things get done by tools that are an amalgam of fragments of theories” (Kent, 1978 p.194).*

## Introduction

Chapters 3 and 4 explored the need to recognise that there was a requirement for both analysis and design as clearly separate activities within the conceptual data modelling process. The roles of the infological and datalogical aspects of a conceptual model, and their required properties, were clarified. The general approaches of E-R and NIAM were described in Chapters 5 and 6, and Chapter 7 provided a comparison of the major differences between the two. It was suggested that the NIAM-CSDP was essentially an analysis tool while the E-R or E-R/R approach had a number of characteristics that lent it to design activities. Finally, Chapter 8 demonstrated the confusion surrounding the definition of a conceptual data model and the lack of consensus among researchers on what should be considered desirable characteristics. In chapter 8 it was also observed that the various modelling approaches are generally considered, and compared, as alternative rather than complementary methods. Taking cognisance of these issues, a framework based on an integration of elements of the NIAM-CSDP and E-R/R approaches to conceptual modelling, is now proposed.

## Previous Proposals

In 1984 Bouzeghoub and Gardarin reportedly developed an expert system, SESCOI, “to support requirements collection in natural language followed by logical and physical design of relational database applications” (Ram 1995 p.97). Kent (1983) has also explored along similar lines. With no apparent knowledge of NIAM or the early work which led to its development, Kent (1983) proposed a method which suggested the identification of facts, defined ‘as connections between things’ as a more effective

method of creating a datalogical structure. One perceived advantage of this method was that no arbitrary decision was required early in the process to distinguish between 'entities' and 'attributes'. Thus both 'Employees have names' and 'Employees are assigned to departments' qualify as facts or more specifically 'fact types or fact patterns', and there is no requirement to allocate any of the objects to a specific type of construct. Having identified the facts that the database is required to maintain, the facts are grouped together in a semi-intuitive way whereby all binary facts that have one object in common are placed in 'pseudo-records' that eventually become relations. Provision is also made to accommodate ternary facts and also for 'facts about facts'<sup>1</sup>.

Kent's (1983) proposal does not appear to have influenced the most commonly used E-R approaches; nevertheless it is significant for two reasons. Firstly, it highlights and discusses a number of shortcomings of E-R modelling as an analysis tool and, secondly, it suggests the use of natural language facts as a means of overcoming some of these problems. However, the method is not "developed to the point of a detailed procedure" and the article primarily "describes the concepts on which the methodology is based" (*ibid.* p.4). One of the shortcomings of Kent's proposed method is that it does not clearly delineate the activities of analysis and design. Kent himself seems rather unclear as to where his proposal sits, saying, "we focus on the middle portion of the analysis and design process" (*ibid.* p.3). Of course, as previously noted in Chapter 4, he is not alone in failing to make any real distinction between the two stages in conceptual data modelling.

It thus seems useful to explore the feasibility of integrating the two approaches of NIAM and E-R/R modelling to determine whether all or any of the described problems can be solved, or at least alleviated, by doing so. Rather than attempting to suggest modifications to any particular system development method, a new approach, the Integrated Conceptual Modelling framework, (INTECoM) is proposed which uses the four-step generic framework of database development, outlined previously on page 27. While the framework covers all four stages in outline, only the first two stages, the

---

<sup>1</sup> These are represented in NIAM as objectified fact types, i.e. fact types that themselves participate in other facts

activities of which have been the focus of the preceding discussions. The discussion of the framework also focuses solely on the design of relational databases.

## **INTECoM - An Overview**

### ***Step 1. The analysis of user requirements***

It has already been suggested that analysis is concerned with determining and describing the components of something complex. Regardless of the viewpoint of the analyst, there is a significant element of discovery, or more specifically 'uncovering', about the activity. When dealing with a specific user, the analyst is required to behave in a largely objectivist way to reveal the perceived information requirements of that particular user. A record of those information requirements is required that it is comprehensible to, and verifiable by, both the user and the analyst, and ideally to others, perhaps at some point in the future. The documented account of the information requirements is the infological or analysis model, referred to in some methodologies as the 'requirement specification'. Ideally, the process followed to extract the user requirements should be predictable and repeatable i.e. any analyst given the same task should arrive at the same result. In other words, a prescriptive method is preferable and this also has the advantage of being auditable. In addition, the analysis model needs to be consistent, unambiguous and transformable into a data structuring representation with no loss of validity and, thus, provide a solid foundation from which to begin design.

The NIAM-CSDP provides a procedure that largely meets these criteria. It has been shown to provide a prescriptive method, which requires the active involvement of the user in providing both the facts and the examples. The direct correspondence between the ORM diagram that is constructed and the formalised natural language example fact types from which it is derived, allow for different representations suitable for either the technical or non-technical user with no information loss. Indeed, as Sharp (1994) argues, non-technical users need never see, or even be aware of, the graphical representation. However, the completed model is transformable, by application of a published algorithm, into a normalised relational design. Most importantly, the requirements can be documented with minimal design decisions having been made, as it

is not necessary to decide on the type of construct<sup>2</sup> that will be used to represent an object before the object, or any facts in which it participates, can be recorded.

The use of a CASE tool, such as *InfoModeler*<sup>TM</sup>, simplifies the collection and maintenance of the natural language facts and the appropriate example set and automates much of the diagram construction. *InfoModeler*<sup>TM</sup> can utilise the entered examples to determine many of the necessary constraints and will highlight situations where the analyst has overridden the determined constraints such that they no longer match the examples. In addition *InfoModeler*'s<sup>TM</sup> comprehensive 'Verbalizer' reports provide all the fact types sentences together with detailed descriptions of all of the objects, at any point during the analysis. *InfoModeler*<sup>TM</sup> also provides facilities for checking syntax, constraints and examples and prevents the generation of a logical model if syntactical problems exist. Finally, *InfoModeler*<sup>TM</sup> is able to automatically apply the transformation algorithm and provides what it terms a 'logical model' in IDEF1X<sup>3</sup> notation.

Darke and Shanks' (1995c) legitimate criticisms of NIAM-CSDP as a tool for requirements elicitation are not addressed by the above description and it is possible that Step 1 of the CSDP could be enriched, as they suggest, by both extending the range from which example sentences are derived and the construction and integration of stakeholder viewpoints (Darke & Shanks, 1994a; 1995a; 1995b; 1995c). There does not seem to be any inherent reason why the NIAM-CSDP could not accommodate both of these suggestions. Indeed, the "assumption that as only one interpretation exists, only one user expert is necessary" (Darke & Shanks, 1994b p.5) is a result of the way in which the method has been used rather than an essential prerequisite of the techniques. Neither is there any inherent reason why the CSDP has to be confined to moving "directly from input and output documents to verbalisation" (*ibid* p.4) nor why conflicting views and alternative viewpoints cannot be explored. If the results of the

---

<sup>2</sup> This point is arguably not as clear-cut as portrayed here. In most version of ORM it is necessary to decide whether an object is an 'entity type object' or a 'value type object', sometimes called a LOT or a NOLOT. However the important point is that no matter which is chosen by the analyst, the functionally appropriate construct will be automatically chosen at the point of transformation.

<sup>3</sup> IDEF1X is a form of E-R modelling widely used in the US and particularly by the Department of Defence.

NIAM-CSDP are seen as a precursor to design and not as the design itself, the pressure to resolve these differences is largely removed. Darke and Shanks' (1995b) proposal to input the integrated viewpoint, developed during the wide-ranging requirements elicitation phase, to the fact type transformation process could in fact be postponed until after the transformation. In this way, a record of each user's requirement as determined during analysis is preserved as discrete documentation. This could prove useful as a basis for the construction of the external schemata, i.e. Step 4 of the database design process. In addition, the resolution of conflicts becomes firmly a design issue, which is where it more appropriately belongs.

The output of the INTECoM analysis phase is a record of users' data requirements, represented as both a set of formalised natural language sentences enhanced with examples and constraint information, and a diagrammatic representation. This final record may be an integrated view or a collection of individual user views. However, whichever it is, each discrete record should be internally consistent, unambiguous and as complete as possible. It should also have been verified, by the users, as accurate and understood. This model is the conceptual model of the meta-model discussed in Chapter 3.

### ***Step 2. The design of the conceptual schema or logical model***

Design has been previously described as the combination of elements into a plan or scheme that conforms to appropriate functional or aesthetic criteria. Thus the design activity is one of creative, and possibly innovative, construction. It is an attempt to bring together possibly conflicting and disparate elements into a harmonious, and ultimately, useful whole. As such, the activity is difficult, if not impossible, to prescribe, relying as it must on the individual flair and creativity of the designer, who will almost certainly bring past experience and experimentation to the work. Attempts to constrain this creativity by mechanistic prescription are likely to be counter-productive. However, the designer needs to have clearly defined elements to work with. An understanding of the required data structuring paradigm is an essential pre-requisite, as is a clear idea of what is required, i.e. the user requirements. The final output of this step will be a datalogical or design model of the data structured in a form that is appropriate to its target DBMS, e.g. a relational model. In other words, the design

model is a paradigm model within the meta-data architecture illustrated on page 35. This output is likely to appear, to the users, to be significantly different from the previous one and while it is impractical to insist that the method used to create it is auditable, nevertheless there needs to be some means of verifying that the original requirement specifications are still being supported.

The E-R/Relational hybrid approach, has been seen to provide techniques which are appropriate to this kind of design activity, at least where the target DBMS is a relational one. It is an inherently creative tool allowing for the development of alternative data structures, which can embody different levels of business rules and constraints. Many of the structures will be those suggested by the patterns identified within the relationships of the data elements themselves but new patterns can be constructed or existing ones enhanced to provide innovative solutions. Used specifically as a design tool, the propensity for entities to be equated with relations is no longer problematic while the need for the designer to make an informed choice of construct for any specific element is no longer dangerous but to be positively encouraged. By the same measure, the requirement for entities to be strictly typed is no longer a cause of difficult communication between the user and the analyst/designer. Instead, it can become a positive advantage to the designer who is now concerned with identifying entities that can be transformed into their strictly typed counterparts within the relational model. In design, there is no longer any expectation of one 'correct' answer but instead an expectation of a number of useful solutions all of which will have their own advantages and disadvantages. Likewise, decisions as to which part of the system will handle each business requirement, can have a direct bearing on the form of the data model (Simsion, 1994), and belong more properly to the designer rather than to the analyst. After all, understanding the compromises and trade-offs involved in the final choice is part of the designer's skill.

Many CASE tools support the use of the E-R/R hybrid techniques, which are promoted here, and a review of these is beyond the scope of this discussion. Even *InfoModeler*<sup>TM</sup> provides some support. The logical model, which *InfoModeler*<sup>TM</sup> creates is an E-R/R hybrid model and can be modified in keeping with the design activities suggested above. As *InfoModeler*<sup>TM</sup> does not support any form of functional or behavioural modelling it may be not be seen as sufficient by the wider development community. In that case the

logical model can be turned directly into a database schema which can be input into any CASE tool which supports the reverse engineering of logical models from database specifications.

The suggested input into the design stage is the output of the analysis phase, described above, which is immediately transformed into a relational logical design. This may be one integrated design or preferably a number of discrete designs, representing the individual user views, for integration by the designer. Design thus begins with clear statements of user requirements, represented in a form that is both familiar and appropriate to the designer. The only requirements which may not have been included in the analysis phase are those pertaining to the less concrete areas such as those which arise from future expectations of the system. These requirements, being only possibilities may not have been identified or fully captured during analysis. An essential ingredient of the design model has to be the flexibility to adapt to future possibilities and the designer needs to be aware of and prepared to incorporate these. Apart from these unknowns, the designer is able to gain a holistic view of the system's requirements relatively quickly and, if the analysis has been carried out competently, with an assurance that no nasty surprises await discovery at a later stage in the process. Thus the development of alternatives and possibly the creation of exploratory prototypes can probably begin early in the design phase. The output of the design stage is a data model conforming to the appropriate paradigm constraints (i.e. normalisation) and ready for transformation to a physical database schema. Its form will thus conform to the usual expectation of the E-R/R hybrid approach, that is an E-R/R diagram<sup>4</sup> supported by the usual data dictionary documentation.

This final design model may well be unrecognisable to the users who provided the initial specifications, yet it is essential that they are able to judge that, despite the resolution of conflicting requirements and incorporation of future possibilities, their requirements can still be met. It appears that, once again, the situation requires users to understand and verify design documentation. This is obviously not acceptable and any proposal that leads to this endpoint is unlikely to hold any advantages over its

---

4 As supported by most CASE tools and practitioner methodologies. It is specifically not the Chen standard.

predecessors. Formalised natural language has been advocated as the preferred user representation for the analysis model and it is proposed that the design model follow a similar course. Chapter 10 describes a method for extracting NIAM type sentences from an E-R/R model to provide not only an understandable translation of the design specification but also a means of linking the design model directly back to the original user requirements.

### ***Step 3. The design of the internal schema or physical database***

This phase would seem to be much less problematic, with many textbooks and industrial courses providing a standard approach (e.g. Connolly *et al*, 1995; Date, 1995; Ricardo, 1990). While a detailed discussion of this step is outside the scope of this study it is nevertheless interesting to note that it is generally recognised as having many of the characteristics of a design activity, but one clearly bounded by certain parameters. Many of those parameters are determined by the technical and environmental constraints imposed by the specific DBMS and implementation environment in which the design must work. Thus there are clear functional criteria to which the design must conform. However, it is also accepted, that beyond some possible denormalisation decisions required by specific performance considerations, the physical design should not deviate in any fundamental way from the structure of the input logical or conceptual model. This is in contrast to many of the previously quoted guidelines for the use of the E-R and E-R/R approach, which begin with an almost clean slate and are guided by no functional criteria apart from the need to produce a communication aid and to avoid any implementation bias. Database designers are also expected to bring their past experience to assist in solving current problems and are also rewarded for innovative solutions, which nevertheless conform to the constraints, placed on the design.

### ***Step 4. The creation of the external schemata***

This step addresses the need to reproduce the original views for each individual user and is usually viewed as merely a task within the previous activity. It is recognised here as a separate step partly for consistency with the generic framework and partly to highlight a particular benefit of the proposed INTECoM framework. Once again, this step, a responsibility of the database designer and generally based on the outcome of the functional analysis that will have accompanied the system development, is largely

outside the scope of this study. However, it is noted that if the individual user requirements have been captured and recorded independently of the final integration, they can provide an important input into this final step. A comparison, between the sentences verified by each user during the creation of the analysis model and the user views created by the database designer, could provide a more direct way of ensuring that the users' views are complete and appropriate.

## **INTECoM - Details**

Figure 14 provides a diagrammatic view of the INTECoM framework. The diagram highlights the inputs from each agent in the process, the outputs produced in each stage and the boundaries between the stages. It is recognised that the framework is not freestanding, in that there are both inputs and outputs related to other aspects of the system development life cycle, which have not been included. This omission is deliberate in that these aspects have not been considered in this study.

### **The Agents**

The human agents, denoted in the diagram by heavy round-cornered boxes, are, superficially at least, self-explanatory. The intention is that these agencies are seen to represent roles rather than particular individuals. It is thus possible for one person to play more than one role in any given project. Where this is the case, the different roles should be viewed as discrete and clear boundaries should be drawn between the activities attached to each role. However, ideally, the roles will be played by different people and the framework, as it is described here, seeks to draw a clear distinction between the data analyst, the data designer and the database designer. It, therefore, presupposes that they are different individuals, with different skills and different responsibilities.

#### ***1 Data Analyst***

The chief responsibility of the data analyst is to record as accurately and as completely as possible the information requirements of the users of the proposed system in a form that is both acceptable to the user and the data designer. It is not necessary, although it is undoubtedly useful, for the analyst to have any deep system or enterprise knowledge,

nor is it necessary for the analyst to be familiar with any specific form of implementable data structures, e.g. relational theory. In fact it is possible that such an understanding, particularly if it stems from considerable experience, could be a hindrance, in that there may be a natural inclination to look for relationally useful structures from the start. It is thus envisaged that the data analyst may not have a background in the more technical aspects of information systems but may instead come from an area such as business analysis.

## ***2 Data Designer***

The chief responsibility of the data designer is to create a data structure that supports the user requirements as documented by the analyst. The design should provide sufficient flexibility to allow for possible future requirements and be transformable into a suitable implementation. It is essential that the designer has a thorough understanding of the implementation paradigm of the target DBMS although knowledge of the specific DBMS is not required. It is envisaged that the data designer would be familiar with data administration issues such as the organisation's corporate data policies and strategic plans as well as having expertise within the data design area.

## ***3 Database Designer***

The role of the database designer is well described in a number of standard texts (e.g. Connolly *et al.*, 1995; Date, 1995; Kroenke, 1992; Ricardo, 1990), and it is not envisaged that the role within this framework will deviate from this.

## ***4 User***

The role of User or stakeholder will typically be taken by a number of individuals drawn from all sections and levels of the organisation, including possibly the technical IS area itself, which have an interest, or 'stake' in the database under development. These sections may include managers, different end-user groups and systems development professionals. Citing Robinson and Bannon (1991), Darke and Shanks describe how "each of these groups may have their own viewpoints and perceptions reflecting their particular organisational context, including their work objectives, work practices, terminology and traditions, and their own set of meanings attached to these" (Darke & Shanks, 1995b p.1). The users' main responsibility lies in their provision of information

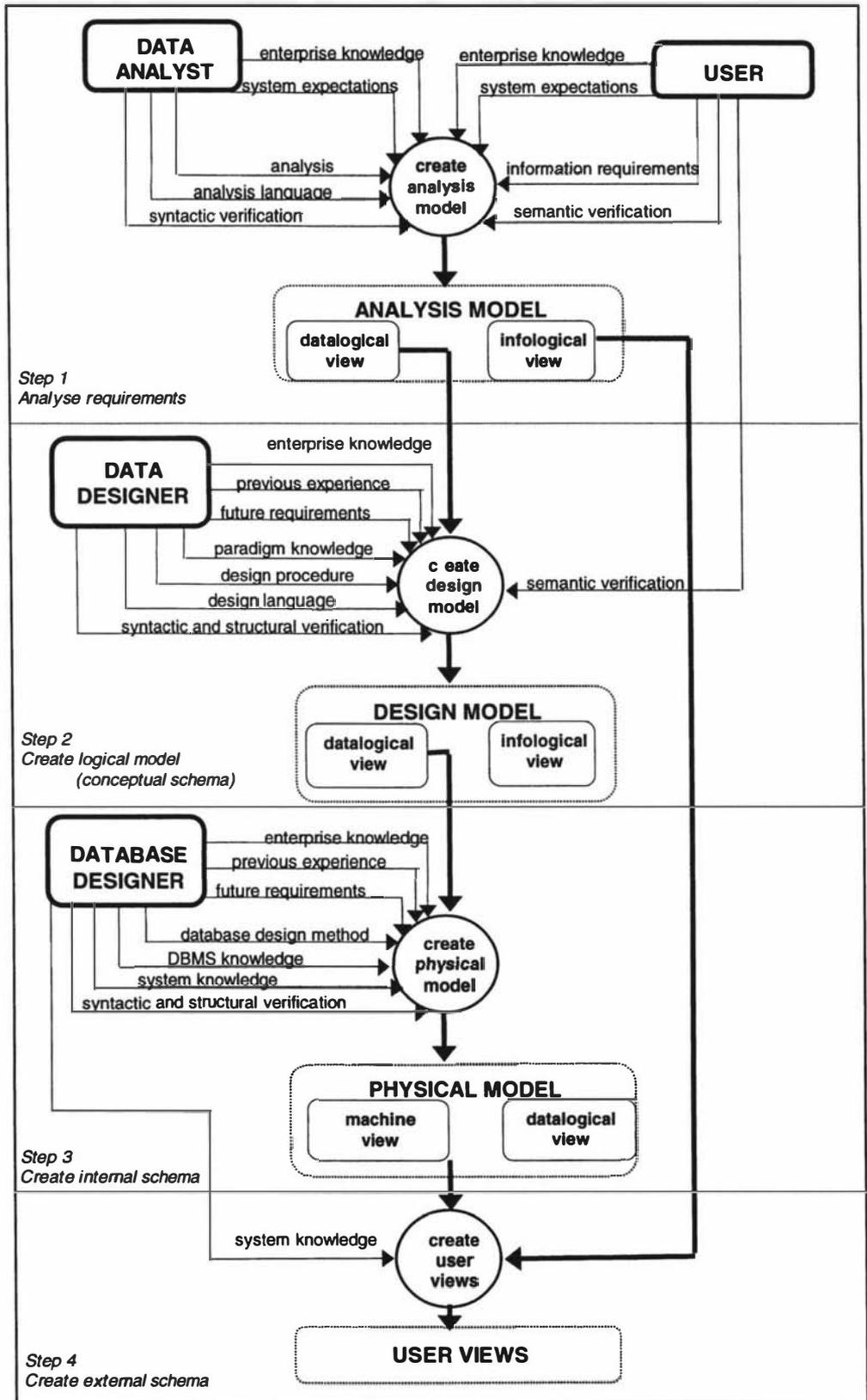


Figure 14. An integrated conceptual data modelling approach

requirements in a form suitable for the analysis process. This will include sufficient knowledge to be able to provide the basis for the construction of facts and examples. It is not necessary for the users to know or understand any form of implementable data structures, nor to read diagrams or graphical representations.

## **The Processes**

The processes, generalised to one for each step for the purpose of diagram clarity, are denoted by circles. The inputs, required by the processes and provided by the human agents, are denoted by annotated light arrows. The outputs, which may themselves act as inputs to other processes, are denoted by heavy arrows. These inputs and outputs will be described in relation to their associated process and instantiated where appropriate.

### **Process 1. Create analysis model**

#### *Process 1. - Activities*

The activities pertinent to this process are those laid down in the seven steps of the NIAM-CSDP discussed in Chapter 6 and are, therefore, not addressed again here. However, it is important to re-iterate that it is envisaged that analysis is not restricted to one user or to only pre-existing input and output documents.

#### *Process 1. - Inputs*

##### *1.1 Enterprise Knowledge*

The first process requires input from both the users and the data analyst. At the general level both agents will bring to the process a level of knowledge and understanding of the proposed system, the system it is replacing or the strategic need it is filling, and the organisational context in which this is to occur. The level and breadth of knowledge will differ both between different users and between the users and the analyst, possibly ranging from the very detailed understanding of a specific area of the organisation's operations to a high-level view of the organisation's plan for the future. The analyst will also gradually develop a view of the internal coherence of the system under scrutiny and possibly from previous systems, a view of how this system integrates with others.

### ***1.2 Systems Expectations***

Both the users and the analyst will also bring a number of expectations of the system to the process. These expectations will include expectations of the process itself as well as of the proposed system. The analyst will have a broad view of the intended scope of the system from which the boundaries of the analysis can be drawn and is also likely to have an understanding of what type of information can be most effectively and appropriately recorded. The user's expectations of the process may range from negative feelings of hostility (Lyytinen and Klein, 1985), or apprehension to positive feelings of enthusiasm.

### ***1.3 Analysis Procedure***

The intention of the integrated framework is that a procedure, having the major elements of the NIAM-CSDP, will be followed in this process and consequently the knowledge and skills relevant to it will be brought to the process by the data analyst. This will include knowledge of suitable CASE tools that can be employed in support of the method as well as techniques for assisting the user in identifying relevant requirements. It is possible that a user too may have acquired some of this knowledge through having participated in previous analysis activity but while this is valuable it is not a prerequisite. Beyond an understanding of what the analyst is attempting to achieve and how that can be facilitated, the user need have no familiarity with the analysis procedure.

### ***1.4 Analysis Language***

If the procedure adopted is NIAM-CSDP then, the analysis language required by this process will be the combination of ORM diagrams and NIAM example fact types. As these two languages are two interchangeable views of the same analysis model, it is unnecessary for the user to understand the ORM graphical notation. However, the analyst needs to understand how to construct appropriate, well-formed fact types as well as how to transform these into the correct diagrammatic representation.

### ***1.5 Information Requirements***

Each user is responsible, assisted by the data analyst, for providing an account of their information requirements to the process. These may be in the form of input and output documents used by the current system but are just as likely to include verbal statements

of clarification about the system or a 'wish list' of currently unfulfilled requirements. It is also essential that the users provide a set of valid examples of the required data. Where possible these examples will be taken directly from the UoD but where not available the user may be required to provide imaginary examples instead.

### ***1.6 Syntactic Verification***

The final model will be checked for syntactic correctness. The NIAM-CSDP contains a number of quality checks, which are largely syntactic i.e. they ensure that the language is being used correctly and unambiguously. As the 'expert' in the language, the analyst needs to provide this input. For example, it is necessary to check that every role has a uniqueness constraint and that the constraints, as stated in the set of examples, are depicted accurately in the diagrammatic notation. Where a CASE tool is being used much of this checking can be performed automatically either as the information is being input or by activating the 'validate' function prior to relational transformation.

### ***1.7 - Semantic Verification***

The user is required to provide verification of the semantic content of the model, e.g. that the fact types and example sentences recorded in the model are meaningful and accurate within the problem domain appropriate to that user.

### ***Process 1. - Output***

The output from this process is the analysis model, which may also be termed the requirement specification. Using NIAM-CSDP, this model is envisaged as a standard NIAM deliverable consisting of both the fully annotated ORM diagram together with the complete set of natural language fact types, object descriptions and examples. The ORM diagram, while not at this stage containing any implementation data structures, is termed the datalogical view as it will provide the basis for the relational transformation and is intended as a step on the way to the creation of the database. The natural language view is termed the infological, as its primary function is to be the communication medium between the analyst and the user. It is this view that is verified and 'signed off' by the user.

The analysis model may either be an integrated collection of user views or may be, as previously recommended, the set of all user views recorded separately. Whichever form

it is in, each user should only be required to verify their own set of requirements in line with Krogstie *et al.*'s (1995) proposal of 'audience-domain appropriateness'.

## **Process 2. Create design model**

### ***Process 2. - Activities***

There are a number of activities within this process, not all of them part of the usual E-R approach discussed in Chapter 5. They can be broadly grouped into three main areas of activity: preparing the first draft of the design, generating and evaluating alternative solutions and creating and verifying the final model.

The preparation of the first draft of the design model will involve the transformation of the analysis model (or models) into an equivalent relational representation. If not previously integrated, the resulting individual models will need to be integrated and conflicts resolved. The requirements embodied in this integrated model will need to be checked with existing databases and models to see whether they can be met from other sources and they may also need to be checked for conformance with elements such as a corporate data model or the organisation's data policy. Finally, the developing model needs to be checked against those of any existing systems with which the proposed system will need to interface. Any necessary adjustments identified from this activity need to be made. At this point the data designer has a holistic view of the user requirements of the proposed system together with any additional constraints or extensions necessitated by the data context in which the system will be finally implemented. The model at this stage can be seen as the initial design model.

The second area of operation is very similar to the more usual activities of the E-R/R approach and will not be discussed at length here. However, it includes the incorporation of future possibilities and in general is concerned with generating and evaluating alternative data structures, deciding on how business rules and constraints can best be supported by the model and on producing suitably normalised entity/relations.

The last set of activities is concerned with the creation of the final design model. Once suitable structures have been designed they need to be checked for syntactic and structural validity. All the data elements included in the model need to have accurate

descriptions and properties recorded and the datalogical view should be extended by the addition of any information required by the database designer, e.g. entity volumes. The data designer also needs to create the infological representation of the final design and assist the user in verifying that the necessary requirements have been met.

It is probable that much, if not all, of the datalogical view will be recorded in a suitable CASE tool. However, at present, there does not appear to be any such tool that will generate an infological view similar to that proposed here. It is thus likely that this will need to be created and maintained as a separate document.

## ***Process 2. - Inputs***

### ***2.1 Analysis Model - datalogical view***

The datalogical view of the analysis model, algorithmically transformed to a relational data structure represented in E-R/R notation, is the base document for the design phase. Once individual views are integrated, it provides a formalised and approved set of user requirements, which must continue to be supported through any subsequent additions, or restructuring of the model through the design process.

### ***2.2 Enterprise Knowledge***

Together with the general understanding of the organisational context shared with the data analyst and users, the designer should also bring knowledge of the corporate database to this process. This information should include knowledge of the data designs of previous systems, standards governing the use and recording of organisational meta-data and areas of data overlap between the proposed system and other organisational databases. In addition, the data designer should have a holistic view of the organisation's data strategy and future goals. While remaining faithful to the specified requirements provided by the previous phase, this knowledge may well impact on the final logical design of the model. For example, data may already exist in an implemented database to satisfy some requirement identified during analysis. To avoid redundancy the new data structure may be deleted from the design model and the relevant data structures from the existing system incorporated instead.

### ***2.3 Previous Experience***

As has been discussed elsewhere in this study, the data designer will bring previous experience to each new data model. This may include models of similar scenarios that the designer has been involved with, patterns of commonly found data structures or experience with structures that have not proved effective as physical databases.

### ***2.4 Future Requirements***

The data designer is probably best placed to incorporate data structures, or the flexibility to include data structures, in support of possible future requirements. This designer will thus bring to the process an understanding of the strategic direction of the organisation and possibly future plans for the proposed system. In addition, previous experience may provide insights into the future possibilities of the system, allowance for which can be incorporated into the data design.

### ***2.5 Paradigm Knowledge***

It is important that the designer has a good knowledge of the data structure paradigm of the target DBMS. Where this is a relational database, then an understanding of relational theory and normalisation is an essential element in the design process. A transformed ORM diagram, for example, will be in 5NF and its further development and design must conform to relational rules if it is to be a useful input into the next phase.

### ***2.6 Design Procedure***

The design procedure in this phase is the descriptive E-R/R approach discussed in Chapter 5. However, as the phase has a more formal starting point, it does not require the early analytic elements. However, this is not to suggest that new data elements are not identified, but to highlight that user input should not be required. The design procedure is thus more concerned with the resolution of potentially conflicting requirements, the manipulation of identified data elements into alternative designs and the exploration of alternative solutions.

### ***2.7 Design Language***

The design language has three components; at its heart is the textual relational schema that is represented diagrammatically by use of the extended E-R/R graphical notation.

This framework introduces a further language element; the use of NIAM-like sentences derived from the E-R/R diagram. The format of the sentences is identical to those produced during the analysis phase and should include the use of examples taken where possible from the user requirements. NaLER, a method for constructing these sentences is described in Chapter 10.

### ***2.8 Syntactic and Structural Verification***

It is the data designers responsibility to verify that both the syntax of the various representations is complete, correct and unambiguous and also that the structures that have been designed, are functionally appropriate and support the requirements specified in the analysis model.

### ***2.9 Semantic Verification***

As with the analysis model, it is important that the users provide semantic verification of the design model. However, they should not be expected to confirm that the data structures themselves are correct, only that their requirements are still able to be met. It is, therefore, important that the design model is available to users in the same form as the analysis model. Provided with the list of example sentences relevant to their view of the domain, it is their responsibility to check that the requirements previously agreed are still supported. The means by which they are met, i.e. the data structures that have been designed to support their needs, should not be a necessary part of this validation.

### ***Process 2. - Output***

The output from this process is the design model. This is envisaged as a standard E-R/R deliverable consisting of both the extended E-R/R diagram and data dictionary together with the corresponding textual relational schema. This is the datalogical view of the design model, which should be suitable for implementation as a relational database<sup>5</sup> although it will not contain specifications of indexes and other physical considerations. In addition to this standard representation, an infological view of the design model will

---

<sup>5</sup> However, in line with common practice, it is likely that some non-implementable constructs, such as subtypes will be included in the design model.

be provided to enable for the semantic verification discussed above. Once again, it is the infological view that is verified and 'signed off' by the user.

### **Process 3. Create Physical Model**

This process is intended to follow the standard activities engaged in by a database designer and described in detail in a number of texts (e.g. Connolly *et al.*, 1995; Elmasri and Navathe, 1989; Ricardo, 1990) and is, therefore, not discussed further here.

### **Process 4. Create User Views**

In general, user views are created in response to the requirements of the concurrent functional analysis and design and it is not anticipated that this will necessarily be supplanted. However, the individual user views as identified and recorded in analysis could be used either to determine the required views or to verify the views suggested by other means. Given the database designer's knowledge of the system and the implemented database, a comparison could be made between the NIAM sentences and the information provided by the various SQL views.

### **Summary**

This chapter has proposed a framework for the integrated use of elements of the existing methods of NIAM and E-R/R modelling. It has suggested that not only could they be used in a complementary way but that their joint use strengthens the overall database development process by providing a much clearer boundary between both the activities and the deliverables of the analysis and design stages. However, with the exception of the techniques for the creation of NIAM sentences from an E-R/R model, detailed in the following chapter, there are no new techniques in this framework. The skill-set required to put the framework into practice is, therefore, accessible to many organisations and, in combination with the one new technique, has a coherence, clarity and consistency missing in many current methodologies. There are a number of implications for both education and practice, inherent in the adoption of this framework and these are discussed in Chapter 14.



# 10 NaLER: completing the circle

*“ ‘How old did you say you were?’  
 Alice made a short calculation, and said ‘Seven years and six months.’  
 ‘Wrong!’ Humpty Dumpty exclaimed triumphantly. ‘You never said a word like it!’.  
 ‘I thought you meant, “How old are you?”’ Alice explained.  
 ‘If I’d meant that, I’d have said it,’ said Humpty Dumpty.  
 Alice didn’t want to begin another argument so she said nothing.” (Lewis Carroll, 1871)*

## Introduction

The INTECoM approach requires a means of translating the datalogical design model into an infological form suitable for verification by users. The technique, termed NaLER, Natural Language for E-R/R, described in this chapter has been developed primarily to meet this need. However, the ability to understand the information content of E-R/R models, i.e. to ‘read’ them accurately, has a much wider application. It is a fundamental skill required by any person involved with E-R/R models in almost any capacity. Not only the modellers themselves and the users whose requirements have been sought, but other end users, such as domain experts, auditors, systems analysts, database designers and administrators, also have a need to ‘read’ a model. It is an important skill for those undertaking teaching and research in data modelling. As Kent (1983 p.51) observes “keeping a record of how the data elements express facts, in terms of the entities and relationships of the business, would constitute excellent documentation of what data is being maintained and what it means.” Despite the increasing need, from an increasing range of people, little attention has been given to it.

Data models have been likened to maps (e.g. Kent, 1978) and the ability to read a data model can be seen to be similar to reading a geographically based map. A map may be consulted by a variety of people for any one of a number of reasons. Each individual user needs to have either a working knowledge of the map symbols or access to a

suitable annotated legend. With this knowledge, the user is able to extract useful information. It seems reasonable to suggest that a person will employ a familiar tool, i.e. natural language, in order to mentally manipulate this new information and will certainly use natural language in order to share, compare or review it with others.

While there is a substantial body of research on map interpretation (MacEachren, 1995), there appears to be little research that investigates how the interpretation of a data model is undertaken. However, it seems likely that a person, attempting to make sense of such a model, will behave in a similar manner to a map user and transform at least some of the semantics into natural language in order to access the new information in the model. Certainly several texts (Simsion, 1994; Veryard, 1984) suggest that analysts may need to occasionally phrase natural language descriptions of all or part of a model in order to facilitate validation of the model by non-technical users. However, there is no indication of how any such descriptions should be derived and certainly very little encouragement to do so. Indeed, Veryard (1984 p.19) states, “most users can be shown the model itself...(as)...there is very little notation to learn and the users need not be bothered with conceptual niceties.” There seems to be an unstated consensus that the interpretation of a data model is straightforward and intuitive once the syntax of both the diagram and the accompanying data dictionary are understood. It is certainly assumed that learning how to build a data model leads directly to an understanding of how to determine what information one contains. This has led some authors to suggest that users may need to be trained more extensively in conceptual data modelling concepts (Hitchman, 1995; Métais *et al.*, 1993). This is analogous to suggesting that only cartographers, or those non-cartographers with extensive cartographic training, can ‘read’ maps. The low level of data model usage, reported by both Hitchman (1995) and MacDonell (1994), would suggest that there is a real problem of comprehension and that data models need to be more accessible to a wider range of people, if their well documented benefits are to be fully realised.

### **The need for semantic comprehension**

Aside from the general desirability of understanding a conceptual data model to assist in tasks such as decision-making and strategic planning, there are a number of specific activities for which a clear and comprehensive understanding of its information content is essential. Firstly, the ‘domain experts’ among the users need to verify that the model

represents an accurate and useful perspective of an organisation's "slice of reality" (Biller & Neuhold, 1978 p.11). Secondly, auditors may use a conceptual data model to confirm that the database that is derived from it processes data accurately and completely (Amer, 1993). The need is not confined to the user community however, technical users of the data model also need this accurate understanding in order, for example, to specify functions against the data, design appropriate physical data structures or assess the impact of a new data model on existing data structures. Thus systems analysts, database designers and administrators and data administrators make up a third group for whom a clear semantic comprehension is essential.

Data modellers, themselves, need to be able to recognise the different semantic implications inherent in alternative conceptual data structures and make informed choices between them (Simsion, 1994). The need for some mechanism of understanding by modellers has also been highlighted by the work of Batra and Antony (1994) in their investigation of common errors made by novice (student) modellers. Their findings suggested that their subjects tended to propose a solution that was based on their initial perception of the problem and that once an initial solution was formulated the basic structure of the model or representation was rarely changed. They suggested that a significant reason for this lack of adjustment to the initial solution was that "since there is no mechanism to inform the designer that the solution is incorrect, there is little motivation to modify the initial solution" (p.64). Batra and Sein (1994), working on the basis of these findings and recognising that "logical database design does not lend itself to self-monitoring by the designer" (p.653), have endeavoured to provide feedback to novice modellers via a design aid called SERFER. SERFER provides feedback by creating a series of structured natural language statements to the data designer based on their input description of a relation. However, the natural language sentences, illustrated in Figure 15, relate not to the information content of the model but to the syntactical rules that are being invoked.

Relationship(s) found :

There is a relationship between :.....**Employee, Skill and Project**

The degree of the relationship is: .....**3 (Ternary)** .....

The connectivity of the relationship is:.....**One-Many-Many**.....

The connectivity is one/many for :.....**Project**.....

The connectivity is one/many for :..... **Employee**.....

The connectivity is one/many for :..... **Skill**.....

An instance each of..... **Employee and Skill** .....is associated with one/many instances of.....**Project**

An instance each of.....**Employee and Project**.....is associated with one/many instances of .....**Skill**

An instance each of .....**Skill and Project**.....is associated with one/many instances of .....**Employee**

**Figure 15. Example of Feedback from SERFER (Batra & Sein, 1994)**

While the notion of timely feedback is valid and seems to offer some fruitful avenues for further research, the nature of the feedback proposed by Batra and Sein (1994) seems rather inappropriate. For novice modellers struggling to come to terms with the differences implicit in various forms of representation, natural language sentences of this form may be confusing and, therefore, less than helpful. While making clear statements about the syntactical elements e.g. ‘The connectivity is one for Skill’, the SERFER feedback does nothing to assist the modeller understand the semantic implications of this syntax i.e. that while a Project may use many Skills, and an Employee may have many Skills, only one Skill may be used on a Project by one Employee. A natural language representation of an E-R/R model, which makes explicit those implications by focusing on the semantics rather than the syntax, would offer a more profitable tool to both modellers and users. The Natural Language for E-R/R models (NaLER) technique, described in this chapter, provides a means for creating such a representation.

### **Uses of the NaLER method**

The creation of the NaLER technique presented here has two objectives. It is intended as an aid to the modeller within the design process and, more importantly, as a means of presenting the information content of the design model to users.

The first objective then, is to provide a self-monitoring mechanism whereby modellers can create their own feedback in terms of the information content of their created structures. Based on the premise that “modelling is essentially making statements in

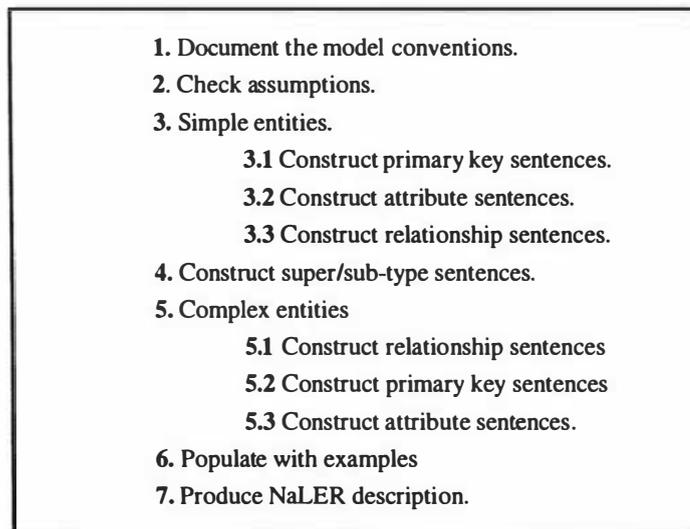
some language” (Lindland *et al.*, 1994 p.43) and that “all representation is an act of knowledge construction” (MacEachren, 1995 p.vii), a useful form of feedback would be to present a natural language translation of the information content of the data structure to the modeller. In this way a designer, in the act of creating a data model, can judge whether a statement created in the modelling language, says what it was intended to say. It is interesting to note that while most texts provide simple translations of simple modelling examples and also rich scenarios to be turned into data models by the students, none provide a rich description of a complex model by way of comparison. However, it is the ability to compare the information within the modelling representation, to the description of the UoD that is most useful in assisting the modeller to understand the implications of the structures that they have created.

The second objective is to provide a means of constructing a description of the modelled world that can be compared with the ‘real’ world. The previous chapter suggested the use of NIAM sentences for semantic verification of user requirements. NaLER provides a means to create similar sentences for the verification of the semantics of the E-R/R design model. In this way, it can be confirmed that user requirements are still being supported.

NaLER also provides a helpful tool for teachers or researchers concerned with evaluating the semantic quality of data models. Chapter 8 showed that almost all researchers involved with data modelling judge the quality of a subjects data model by comparing it with a previously worked ‘correct’ solution. However, this process of comparison has several difficulties, one of which is determining semantic equivalence between two different structural representations. In order to determine whether two facts are equivalent there must be a commonly agreed method of extracting the facts from the data schema as well as an agreed means of comparing them. While their paper describes a fairly complex formal method for comparing the equivalence of different representations, Biller and Neuhold (1978) themselves conclude that comparing the data representation to reality “must rely on a common understanding of natural language” (p.29). As NaLER provides a means of extracting a complete set of natural language sentences from an E-R/R model, any number of sets can be compared either with each other or with an initial set extracted directly from the UoD. Such comparisons can

highlight incorrect or new semantics and provides a genuine basis for assessing semantic equivalence between differing data structures.

The technique could also assist with another problem faced by evaluators of data models; their reportedly low level of understandability. Shanks (1997) observes that data models need to be better explained and he suggests exploring the use of various natural language based techniques such as narrative scenario descriptions (ascribed to Carroll, 1995) and the argumentation-based design rationale (Buckingham *et al.*, 1994). The NaLER technique described here could be a useful addition to these.

- 
1. Document the model conventions.
  2. Check assumptions.
  3. Simple entities.
    - 3.1 Construct primary key sentences.
    - 3.2 Construct attribute sentences.
    - 3.3 Construct relationship sentences.
  4. Construct super/sub-type sentences.
  5. Complex entities
    - 5.1 Construct relationship sentences
    - 5.2 Construct primary key sentences
    - 5.3 Construct attribute sentences.
  6. Populate with examples
  7. Produce NaLER description.

**Figure 16. NaLER - An overview**

Thus, there are a number of situations that would benefit from a natural language interpretation of an E-R/R model and the creation of such an interpretation is a natural and intuitive response to the need to make sense of such a model. The proposed technique, an overview of which is illustrated at Figure 16, is designed to capture this response in an organised way and to encourage data modellers to create a semi-formal natural language description to elucidate the information that their models contain.

### **The NaLER Method**

NaLER is designed for use with a relational data model, represented diagrammatically by an E-R/R diagram and supported by a data dictionary, as produced by many contemporary CASE tools. The more extensive the available documentation, the more effective the translation will be.

It is envisaged that the data designer would use NaLER either to check the semantic content of certain data structures under development or to present documentation to users for verification. The NaLER user will thus have a good understanding of both the relational paradigm and the syntax of E-R/R models. When used by practitioners it is therefore reasonable to expect the following pre-requisites to be met.

- P1 - Entities are named.
- P2 - Entities will have a unique identifier or primary key.
- P3 - Lines between entities denote Primary Key/Foreign Key relationships.
- P4 - Relationships are named in at least one direction.
- P5 - Relationship cardinality is indicated on the diagram.

### 1.1 Assumptions

Although there are a number of additional desirable elements, it is recognised that not all CASE tools provide the same level of documentation support and also, that when used in support of the modelling process, not all the information may have been recorded. Therefore, while full documentation is recommended, if the model is incomplete in some way there are a number of assumptions that can be made. These assumptions are that,

- A1 - Relationships are optional unless clearly annotated as mandatory<sup>1</sup>.
- A2 - A 1-1 mandatory relationship is implicit in the position of an attribute in an entity<sup>2</sup>.
- A3 - If an attribute is described as nullable, the 1-1 relationship is optional.
- A4 - Two attributes with the same name placed in different entities relate to the same 'real world' concept<sup>3</sup>.
- A5 - If any such attribute is a primary key in one entity then it is a foreign key in any others in which it appears.

---

<sup>1</sup>This assumption is based on the findings of Siau *et al.* (1995) who suggest that experienced modellers will almost always prefer to show relationships as optional unless there is very clear evidence to the contrary.

<sup>2</sup>This assumption only holds if the model is in 1NF and assumes that the **intention** of the designer would be, minimally, to create an E-R/R structure in first normal form.

<sup>3</sup> While it is not a requirement that attribute names are unique within a model, it is generally good practice to make them so. Thus if the same attribute name appears in more than one entity it can be assumed to relate to the same 'real world' element. If it becomes clear that this is not the case, the attributes should be renamed to remove ambiguity.

A6 - An entity whose primary key consists of two or more foreign keys is specifying a many to many relationship between the entities of which those attributes are primary keys.

These are the only assumptions that should be required where the data designer who is using NaLER has constructed the model. However, when sentences are being extracted from models created by other designers, particularly students or research subjects, it is possible that a number of syntactic errors can impede the process. In this case some additional assumptions are proposed which, while requiring more judicious use, can assist in maximising the amount of useful information that can be extracted. These are,

A7 - For unnamed 1-m relationships, the parenthesised name '(has)' can be used, unless a more intuitive one is suggested by the names of the participating entities.

A8 - For unnamed 1-1 relationships, the name '(has)' can be used unless the two entities have an identical primary key in which case the name '(is)' can be used.

A9 - If a foreign key attribute exists without an existing relationship line, then the relationship should be treated as missing and unnamed. It may be useful to create the relevant sentence using 'has' as the relationship name and a cardinality of 1-m.

It is suggested that any adjustments that are made on the basis of these assumptions are clearly marked in the final description, either by the use of parentheses as here or by some type of formatting such as bolding or italicising.

## **1.2 Procedure**

The procedure is broken down into 6 steps. For reference, all the statements are numbered as they are constructed, however, the ordering of the statements is not significant.

### ***Step 1 - Identify and document the diagram conventions***

This purpose of this step is to record and clarify what notation has been used to construct the diagram and data dictionary. Where a CASE tool has been used it may be unnecessary although for future reference it is useful to record how the model is being interpreted. Any inconsistent use of notation in the model should also be noted.

**Step 2 - Check what assumptions need to be made**

The purpose of this step is to identify any areas in the model where assumptions from the above list will need to be made. Where this is a check of the designer's own work, it can be seen as a syntax check and it is expected that most of the ambiguities and omissions that are discovered can be resolved. In other circumstances, 'corrections' should not be incorporated. Any assumptions that are made should be recorded.

**Step 3 - Identify each simple entity**

This step is concerned with extracting the sentences that relate to the simple entities within the model by completing the following 3 tasks.<sup>4</sup>

**3.1 Construct a sentence for the primary key attribute(s) as<sup>5</sup> :-**

**Sn: Each <entity-name> is uniquely identified by <primary key>.<sup>6</sup>**

e.g.

**S1: Each Zoo is uniquely identified by zoo-no.**

**S2: Each Zoo-Animal is uniquely identified by zoo-no, animal-no**

This task is intended to focus on the appropriateness of the chosen primary key and the entity name.

**3.2. For each attribute, construct a sentence as :-**

**Sn: Each <entity name> (<primary key >) must have only one <attribute name>.**

e.g.

**S3 : Each Zoo (zoo-no) must<sup>7</sup> have only one Zoo-name**

During this task, a sentence should be constructed for all attributes except those that are primary or foreign keys. This step works on the assumption that the model is in at least First Normal Form. However, this step can also be valuable in checking the level of

---

<sup>4</sup> A 'complex' entity is defined as one which is representing the resolution of a ternary (or higher degree) relationship. All other entities are termed 'simple'.

<sup>5</sup> A formal description of the NaLER language can be found at Appendix 2 .

<sup>6</sup> Italicised words are constants.

<sup>7</sup> Note that if an attribute is described in the dictionary as nullable the relationship is optional not mandatory in which case 'must' should be changed to 'may'.

normalisation by providing an intuitive means of thinking and talking about functional dependencies.

**3.3. For each binary relationship that the entity participates in, construct two sentences as :-**

***Sn:*** Each <entity-name> (<primary key>) <optionality> <relationship-name>  
<cardinality> <entity-name> (<primary key>).

***SnR:*** Each <entity-name> (<primary key>) <optionality> <relationship-name>  
<cardinality> <entity-name> (<primary key>).

e.g.

**S4:** *Each Zoo (zoo-no) houses one or more Animal (zoo-no, animal-no)*

**S4R:** *Each Animal (zoo-no, animal-no) is housed in one Zoo (zoo-no)*

The form of these sentences may be familiar as a variation of the business rules used in some methodologies. However, the inclusion of the key attribute(s) makes a significant contribution to the general understanding of the model. For example, the sentence S4, which clearly identifies that animal-no is not unique across zoos, can prompt the designer to check that the functional analysis recognises that each zoo is responsible for uniquely numbering their own animals. As an extension of this thinking it may also be useful to check what happens if an animal is moved from one zoo to another, either permanently or temporarily.

These sentences are sufficient for the translation of a significant proportion of E-R/R models, however, there are two other types of relationships i.e. the relationships between super and subtype entities and the complex relationships surrounding resolution entities, that these steps handle less effectively. The following two steps are designed for these.

**Step 4 - Supertype/subtype sentences**

**For each subtype entity construct a sentence of the format:-**

***Sn:*** Each <sub-entity-name>(<primary key>) is a <super-entity-name>(<primary key>).

e.g.

**S5:** *Each Mammal (zoo-no, animal-no) is a Animal (zoo-no, animal-no)*

**S6:** *Each Reptile (zoo-no, animal-no) is a Animal (zoo-no, animal-no)*

These sentences should be straightforward to construct and may seem almost trivial. However, it can prove useful to check that the primary key is being correctly inherited and can also provide an insight into the nature of the relationships between the subtypes themselves. For example, the sentences can help to identify overlapping subtypes, which, while not proscribed, may not be desirable (Simsion, 1994).

### ***Step 5 - Identify complex entities***

This is the most difficult task as these entities are not always straightforward to identify. If relationship sentences have already been created and the underlying semantics are clear then they may be omitted. However, if the resulting sentences are not clear it may be beneficial to rework them using the appropriate formats described in this step. For completeness, all the potential sentences should be generated although this is not usually necessary. In practice it is useful to generate only those that contribute to the overall understanding. For convenience, certain abbreviations have been used here, i.e. e = entity, pk = primary key, and r = relationship.

#### ***5.1a. For entities with a composite name, e.g. 'Employee/Project/Skill'***

**Sn:** Each <e-name> ( <pk> ) <optionality> <r-name> <cardinality>  
 <e-name> ( <pk> ) and <optionality> <r-name><cardinality >  
 <e-name>( <pk>).

e.g.

**S7: Each Employee (emp-id) must be employed on one or more Project (proj-code) and may use one or more Skill (skill-code).**

**OR**

#### ***5.1b. For entities with a simple name, e.g. Invoice***

**Sn:** Each <e-name> ( <pk> ) <optionality> <r-name> <cardinality>  
 <e-name> ( <pk> ) and <optionality> <r-name> <cardinality>  
 <e-name> ( <pk> ) and <optionality> <r-name> <cardinality>  
 <e-name> ( <pk>).

e.g.

Given an entity 'Invoice' linking Salesperson, Customer and Order

**S8: Each Invoice (invoice-no) must be made by only one Salesperson (Salesperson-id) and may be for only one Customer (customer-no) and may include one or more Order (order-no).**

There are a number of issues surrounding the construction of these sentences. Firstly, it is important to note that in the situation illustrated in step 5.1a, the entities named are those that **participate** in the relationship rather than the resolving entity itself and that therefore, the primary keys are also those of the participating entities. However, they will also appear as foreign keys (and possibly as part of the primary key) of the resolving entity.

In the second case, shown at 5.1b, the first entity that is named i.e. <e-name> is the resolving entity while the others are the entities that participate in the complex relationship. Secondly, it is probable that other sentences relating to the resolution entity have already been constructed, e.g. to allow for the relationship between Employee and Employee/Project/Skill, or Customer and Order. If this is the case the original sentences should be deleted, as they will represent only a partial view of the underlying semantics.

Finally it should be noted that it might not be possible to use the relationship names provided on the model for these types of sentences. While it is desirable to keep the wording as close as possible to the model it may be necessary to make some adjustments to keep the sense of the relationship.

***5.2 For each complex entity, construct a sentence for the primary key attribute(s)***

This task is the same as for Step 3.1

***5.3 For each complex entity, construct a sentence for each of the other attributes***

This task is the same as for Step 3.2

***Step 6 - Populate the sentences with valid examples***

This step is a reversal of the technique within the NIAM-CSDP whereby 'fact types' are derived from the elementary natural language 'facts'. Here example sentences are generated from the fact-types and instantiated by relevant valid examples taken from the UoD. Where NaLER is being used within a system development process<sup>8</sup>, the examples should be taken from the analysis documentation wherever possible, rather than

---

<sup>8</sup> If NaLER sentences are being generated for an existing database, the examples should be taken directly from the database itself. (Ryder, 1996)

generated in isolation by the designer. Any examples that are created by the designer should be verified by the user. Generally, there is little point in generating examples for the primary key sentences as these are explicit in all the sentences related to the entity holding that primary key

**S2: Each Zoo (zoo-no) must have only one Zoo-Name**

Zoo (111) has the name 'Regents Park'

Zoo (222) has the name 'Whipsnade'

**S4: Each Zoo (zoo-no) houses one or more Animal (zoo-no, animal-no)**

Zoo (111) houses animal (111,123)

Zoo (111) houses animal (111,345)

Zoo (222) houses animal (222,123)

***Step 7 - Produce a full NaLER description***

Beyond checking for inadvertent duplication<sup>9</sup>, this step should not involve any additional work. It brings together all the constructed sentences to create a standardised description. The sentences can be listed numerically, grouped together in a way that is likely to be meaningful to the user or arranged to match the order of the sentences in the analysis model. Taking the latter option is particularly effective when the user is required to confirm that the design model is supporting the original specifications, as happens at the end of the design stage in INTECoM.

Any sentences that were adjusted by the use of the assumptions in Step 2 should be annotated. For example, S4 and S5 in Figure 17 indicate by the bold type that the relationships were optional by default and S4 also shows a missing relationship name. All the examples used in the NaLER description should be valid ones and sufficient; in other words, there should be a direct correspondence between the examples and the constraints explicitly stated in the fact types. Where a discrepancy exists, it must be investigated. Although it may be only an error in the collection of examples, it may be highlighting an incorrect constraint that needs to be adjusted in the model. It is also helpful, although not essential, if the examples are internally coherent and consistent. It is much more understandable, particularly in a complex domain, if one valid example 'case' can be tracked through all relevant sentences.

---

<sup>9</sup> This only applies to sentences that have been duplicated through some error in the NaLER process. Sentences, which have been duplicated in the analysis stage, should not be removed until the necessary changes have been made to the model itself.

S1:	Each Zoo is uniquely identified by zoo-id
S2:	Each Zoo (zoo-no) must have one location Zoo (111) has location "London" Zoo (333) has location "Sydney" Zoo (222) has location "London"
S3:	Each Zoo (zoo-no) must have one Zoo_name Zoo (111) has name "Regents Park" Zoo (222) has name "Whipsnade"
S4:	Each Zoo (zoo-no) <b>may own</b> one or more Animal (zoo-no, animal-no) Zoo (111) owns Animal (111,123) Zoo (222) owns Animal (222,123) Zoo (111) owns Animal (111,345)
R	Each Animal(zoo-no, animal-no) must be owned by one Zoo(zoo-id) Animal (111,445) is owned by Zoo (111) Animal (111,567) is owned by Zoo (111) Animal (222,123) is owned by Zoo (222)
S5:	Each ZooAnimal (zoo-no, animal_no) <b>may</b> belong to one AnimalType ZooAnimal (111,123) belong to AnimalType "Panther" ZooAnimal (222,123) belongs to AnimalType "Kangaroo"

**Figure 17. NaLER sentences and examples.**

## Summary

The NaLER technique can be of benefit in a number of different situations. However, it is primarily for use by data designers, as a means of 'proof-reading' their own work and as an aid in gaining user verification for the design model. In the first case, it is envisaged to be of most use to novice designers, by providing them with some immediate feedback about their datalogical constructions. However, as Kent observes "if a mind is committed to a certain model, then it will perform amazing feats of distortion to see things structured that way" (Kent, 1978 p.93) and expert designers are also susceptible to this. In the second case, it provides a way to communicate the semantic content of a design model to users and removes the necessity for users to understand the language, and the decisions, underlying the datalogical design model.

It is recognised that NaLER does not cover all possible constructs and variations in all E-R methods. However, it is suitable for the extraction of sentences from the E-R/R models most often used in practice, as described in various practitioner texts (e.g. Simsion, 1994).

# 11 INTECoM: in practice

*“[C]onceptual modeling is valuable and I believe that InfoModeler is by far the best conceptual modeling tool. Perhaps a hybrid solution is in order for those who need and can afford it: use InfoModeler for initial conceptual modeling and create a logical model and forward engineer it into a database schema. Then reverse-engineer the schema into a logical model with a better logical- and physical- modeling tool and use that tool to tune your model” Grimes (1998)*

Chapters 9 and 10 described a theoretical framework in which techniques for the analysis and design of data can be integrated. Two well documented, widely used and tested methods were chosen to instantiate the framework, although there is no reason why other appropriate methods could not be substituted. Previous chapters developed a clear case for the advantages offered by employing methods with particular ‘ways of working’ at appropriate stages in a database development. Although, as previously stated, there seemed little to be gained by conducting another comparative evaluation of E-R and NIAM methods, neither the mapping of ‘ways of working’ to developmental stages nor a comprehensive integration of two disparate techniques has been reported in the literature. Clearly, the soundness of the arguments that lead to the development of INTECoM, and the viability of the framework itself, would be strengthened by the successful use of INTECoM in an actual development. INTECoM is primarily designed to bring benefits to non-trivial developments involving a number of developers and users. Therefore, a consequential evaluation of the framework would, ideally, require a substantial development project by a small team of analysts and designers within a medium to large organisation. The level of resources required for such an undertaking places it beyond the scope of this research, which has theory building rather than theory testing as its focus. Nevertheless, a small development, in the nature of a worked example, is useful in demonstrating the feasibility of the framework, its practical application and the validity of the arguments employed in its creation. Consequently such a development was designed, with the researcher as both database developer and

one of the primary users of the intended system. The intention of this exercise was not to observe or evaluate the efficacy of the framework, or its constituent parts, but to test and refine the theoretical basis on which the framework was established. In addition, the example was designed to focus exclusively on the development of the database design and therefore did not attempt to integrate INTECoM with the analysis and design of events and behaviour.

A description of the entire development, which follows the procedures laid out in Chapter 9, is included here and the complete set of deliverables, as well as some additional transitional documentation, is included in the appendices and referenced as appropriate. The development work resulted in a number of useful observations and refinements, which are also discussed here and consolidated into a revised framework in Chapter 13. Quality evaluation of the development, utilising some useful concepts from Chapter 8, was undertaken and this is discussed in Chapter 12.

## **System Development**

The system that was chosen to provide a worked example was the re-development of a database of postgraduate student information (ISPG) required for internal use by the Department of Information Systems at Massey University. This system required the creation and maintenance of an accurate, current record of departmental postgraduate enrolments. The database would be used by the postgraduate co-ordinator to provide timely information to the Head of Department and also to ensure that individual postgraduate programmes were correctly reported to the various College Administrators. It would also provide a facility for *ad hoc* querying by paper co-ordinators and students.

The amount of data that would be stored in the system would be small and it was envisaged that the database would be held on the local server with access restricted to the three or four people who would regularly require information from the system. It was recognised that some of the data would duplicate that held by the University's central computer system. However, for a variety of reasons it was considered expedient to carry the cost of this duplication. A similar system was already in place within the Department. However, it had been developed informally and had grown in scope and complexity since its inception. There was thus a genuine need to revisit the database design. From the context diagram illustrated at Figure 18 it can be seen that the major

information users of the system would be the Head of Department and the College Administrators. The details of the major dataflows are described at Appendix 3.

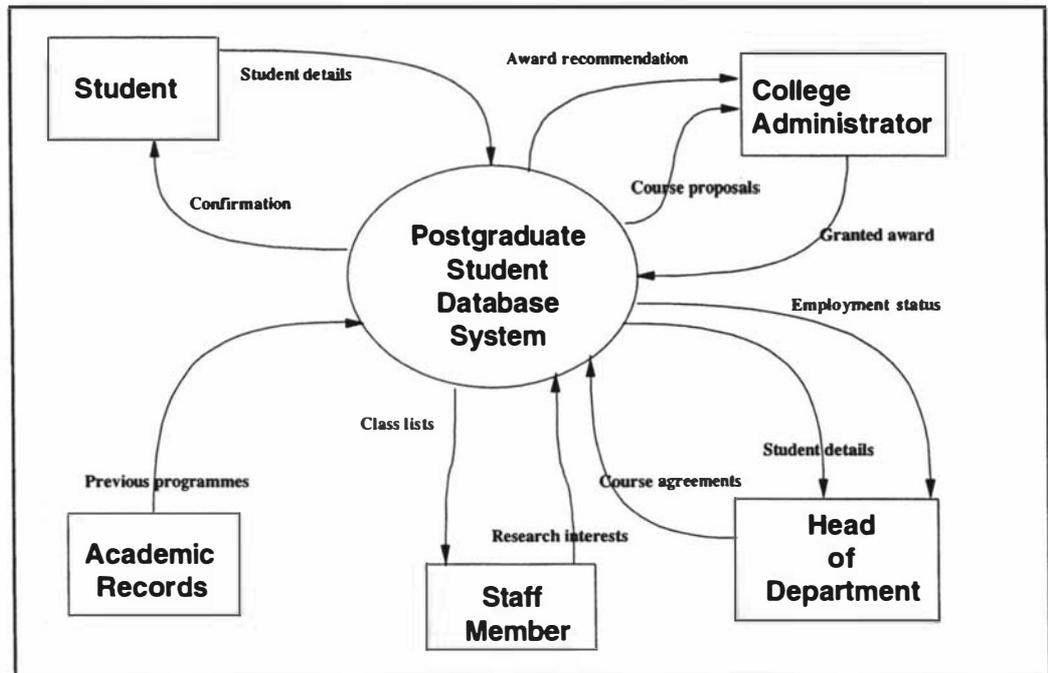


Figure 18 ISPG System - Context Diagram

## 1. The analysis of user requirements.

### 1.1 The Agents

INTECoM begins with the identification of the various agents, Data Analyst, Data Designer, Database Designer and Users, within the system development process. INTECoM encourages the identification of different individuals for these roles but recognises that one person may play more than one role, and this was the case in this development. The researcher played the first three roles and was also an expert user. However, there had previously been significant input into the requirements by a variety of users and other user input was solicited when appropriate and possible.

### 1.2 Inputs

A number of inputs are described as relevant to the analysis phase of INTECoM. In this example, both users' and the data analyst's *enterprise knowledge* was extensive and their *system expectations* realistic and informed. The *analysis procedure* of the NIAM-CSDP and the *analysis language* of ORM and NIAM were well understood by both the

data analyst and some of the users. The remaining domain specific inputs are described below.

*Information requirements.* The information requirements of the system were considered from the following four viewpoints, students, staff members who are paper co-ordinators, the Head of Department (HoD), and the College postgraduate administrators. The requirements were derived from a number of sources including, the researcher's own experience of the current system, examination of current documentation and interviews with users.

*Students* require confirmation that

- any specific decisions agreed with the HoD have been recorded accurately, particularly the content of approved special topics,
- they have enrolled in a program of study which conforms to university, college and departmental regulations, and
- that correct contact details have been recorded

*Paper Co-ordinators* require

- a list of people enrolled in the paper, and
- current contact details of each student.

*The Head of Department* requires

- a record of each student's current programme,
- a record of each student's past programme including grades,
- a record of each student's future study plans, if known,
- decisions agreed with the student,
- the student's employment status, and
- a record of each student's research areas and their research supervisor.

*College Postgraduate Administrators* require

- notification from the department of a proposed course of study for formal approval by the College board,
- notification from the department of completed study with a grade point average score for the complete programme, and
- recommendation that an award is granted with a level, e.g. First Class Honours.

*Syntactic Verification.* The qualified fact types and examples derived during the NIAM-CSDP process were input to InfoModeler™ and all views were validated on completion. This CASE tool will not allow the transformation of the initial ORM unless it is syntactically correct hence syntactic verification was straightforward. However, there is obviously no guarantee that a syntactically correct model is logically correct.

*Semantic Verification.* The developer, as an expert user of the system, conducted the initial semantic verification of the model. However additional verification was also obtained from appropriate users. These users were shown only the qualified fact types relevant to their own view and the ‘Verbalizer’ report from *InfoModeler*<sup>TM</sup> showing the examples<sup>1</sup>. At no time were the users shown the ORM diagram or any potential E-R diagram although most users would have been able to interpret either.

### 1.3 The Analysis Process

The process that was followed was largely that laid down by the NIAM-CSDP, adapted to the effective use of *InfoModeler*<sup>TM</sup>. Each view was considered separately and entered into a discrete model within the CASE tool. The decision was made at the beginning of the process to keep the user views separate during the analysis stage and not to integrate them until the design stage. It was therefore necessary to open a new model for each view because even while the ORM diagrams can be kept on separate pages within *InfoModeler*<sup>TM</sup>, the validation of the models and the transformation to a relational structure will always consider all pages together. As the facts were being constructed, every effort was made to ignore relational considerations and to concentrate only on the sentences that could be derived from the user view. Therefore the qualified fact ‘*Staff (Staff\_code) has name*’ which seems a very obvious and necessary fact to include, does not appear in the analysis model, as it was never identified in the users’ view. Neither was there a conscious effort to create consistency between the user views. While the College Administrator view requires a name for the *supervisor*, this was not equated by the analyst with *Staff\_code*. Another example lies in the apparently conflicting identifying structure for a student’s enrolment in a paper, which is discussed on page 170. The conflict was noted but no attempt was made to resolve it until design.

#### 1.3.1 Student View

Initial sentences were determined as required by the student view of the system through discussion with current postgraduate students and examination of current documentation. A complete list of these sentences is included at Appendix 4.1.1. The student view is typically constrained to one year at a time and is mainly concerned with

---

<sup>1</sup>The qualified fact types and the example sentences are included at Appendix 4.

ensuring that the chosen programme has approval and meets with all the necessary regulations in force at the time of enrolment. Although there are other areas of legitimate interest to the student view, e.g. the timetable for each paper, assignment and examination dates, these were not considered to be of interest to the development. This view was considered to have the least priority. It was recognised that students, in general, operate from a position of trust in the department's ability to counsel them into a suitable and approvable programme. They tended to view the need to confirm their details as a piece of bureaucracy but for those few students for whom it was relevant, the knowledge that the content of any Special Topic was recorded and in line with their own understanding was seen as important.

These sentences were converted to qualified fact types and input to *InfoModeler*<sup>TM</sup> together with appropriate examples using the Fact Assistant facility. Three sentences were not included explicitly as facts. One was implied by the use of the `student_id` as the identifier of a student object and two others were shown as subtype constraints on the ORM diagram rather than entered as fact types. The complete set of qualified fact types, the associated examples and the ORM diagram are also included at Appendix 4.

### **1.3.2 Paper Co-ordinator View**

A set of initial sentences was derived from discussion with some staff members who were paper controllers, from existing reports and from the personal knowledge of the developer (who was also a paper controller). A complete list of these sentences is included at Appendix 4.1.2. While staff members sometimes required much more detailed information about students, this was generally available from the University's academic record system. The information requirements represented here were thus restricted to those that were currently produced by the departmental system. This view was accorded third highest priority. The most common use for this view was to produce an accurate list of students enrolling in a paper, as this was not always available from the central university system at the very beginning of a semester. It was also used to provide a list of students who had enrolled early for the next semester. However, it was also used to report a list of students enrolled in Special Topics by the postgraduate co-ordinator, as the staff member nominally responsible for these papers.

These sentences were also converted to qualified fact types and input to *InfoModeler*<sup>TM</sup> with appropriate examples. As *InfoModeler*<sup>TM</sup> is restricted to a maximum of a quaternary fact, two nested fact types were created from these sentences, *PaperSemesterYear* and *StudentPaperSemesterYear*. The first represented the offering of a particular paper in a particular semester, which may have different characteristics from another, e.g. it may be taught by a different staff member, or may have a different points value. The second nested fact type represented the enrolment of a student in a *PaperSemesterYear*. A decision<sup>2</sup> was made which was to add a property of *Type* to the object *PaperSemesterYear* to identify whether the paper was a research project, a special topic or a taught course. This type was implied in sentences 12 and 13 although the same information could have been represented as showing *SpecialTopic* and *ResearchProject* as sub-types of the *Paper* object as it was in the Student View. Finally, although not explicitly stated in the initial sentences, *Supervisor* was shown as a sub-type of the *Staff\_member* object<sup>3</sup>. The complete set of qualified fact types, examples and the resulting ORM diagram are included at Appendix 4.

### 1.3.3 Head of Department View

A set of initial sentences was derived from discussion with the Head of the IS Department, from existing reports and from the personal knowledge of the analyst. A complete list of these sentences is included at Appendix 4.1.3. The Head of Department had three major areas of focus. One was the overall postgraduate programme of an individual student, another was the allocation of staff and the third was the research profile of the department. The original system was created in response to the information needs of the Head of Department and this view was consequently given the highest priority.

These sentences were converted to qualified fact types and input to *InfoModeler*<sup>TM</sup> with appropriate examples, using the Fact Assistant. Three nested fact types were derived from these sentences *StudentProgramme*, *StudentProgrammePaper* and *StudentEnrolment*. The first represented the enrolment of a student in a particular

---

<sup>2</sup>The inappropriateness of making a design decision at this early stage was not apparent until much later.

<sup>3</sup>Again an inappropriate design decision was made which caused later problems. There was no indication that a supervisor was a staff member except for the developer's own presumption that this was the case.

programme, e.g. a PhD, which might last over an indefinite time period. The second represented the inclusion of particular papers in a programme and the last represented the enrolment of the student in a particular paper in a specified semester and year. Supervisor is again shown as a sub-type of the Staff\_member object. The complete set of qualified fact types, examples and ORM diagram are included at Appendix 4.

### 1.3.4 College Postgraduate Administrator View

The sentences for this view were derived exclusively from the existing documentation that is sent to the various Colleges, several examples of which were available to the developer. Further confirmation was not sought from the administrators as the requirements were well known and had been stable for a number of years. Any change in requirements would be notified to the department.

These sentences, included at Appendix 4.1.4, were also converted to qualified fact types and input to InfoModeler™, together with appropriate examples. Two nested fact types, CompletedPaper and StudentProgramme were derived from these sentences. The first represented a paper taken by a student in a particular year that was to be credited towards the completion of a particular programme. The second reflected the student's enrolment in a particular programme. Three sentences, 7, 11 and 12 appeared to be derivative and were recorded as within the CASE tool.

### 1.4 Final Semantic Verification

The developer initially checked all the fact types and examples for semantic correctness and completeness. This entailed ensuring that every sentence had been recorded within the ORM, either as a qualified fact type or as a constraint (*completeness*). The example sentences were then checked to see if they conformed to the analyst's understanding of the problem domain (*correctness*).

A representative of the first three user groups were then shown both the initial sentences and the example fact types and were asked to comment on them and confirm their accuracy. A number of changes were made at this point. Although most were trivial the student user commented that students would also want to know the points value of each paper, the total points required by a programme and the name of the person who was running a paper. These three additional facts were added to the Student view.

## 1.5 The Output

The successful completion of the process so far resulted in a set of documentation that INTECoM calls the analysis model. The infological view, i.e. the complete set of natural language fact types, object descriptions and examples, which have been verified by the user, can be viewed as the complete requirement specification for the data aspects of the system. In contrast the datalogical view, i.e. the ORM diagrams and repository information that would provide the input for the relational transformation of the design phase, had so far played almost no role in the process. However, because the datalogical view was being created synchronously with the fact types and examples, the logical relational schema, which would eventually be produced, could be taken as preserving the necessary semantic equivalence between these two forms of the model.

## 2 The design of the logical model

### 2.1 Inputs

Once again, not all of these inputs will be discussed in detail. The comments on *enterprise knowledge* made in the previous section are again relevant. The data designer brought *previous knowledge*, *paradigm knowledge* and an understanding of the *future requirements* of the system to the design process and was familiar with all aspects of the *design language* for this stage, i.e. a textual relational schema, an E-R/R diagram and NaLER sentences. The remaining inputs, the *analysis model*, the *design procedure*, and the various *verification* requirements will be discussed as they become relevant.

### 2.2 The Process

#### 2.2.1 Prepare the first draft of the design.

The initial activity in the design stage was the transformation of the datalogical analysis model into an equivalent relational representation. For each view, a logical model was generated automatically by InfoModeler™ and a note made of the initial layout of the ER/R representation. The awkward names of some of the generated relations and attributes were changed but no alteration of any kind was made to the data structures themselves. InfoModeler™ was then used to create a generic SQL schema for each view. The most efficient means to create the initial E-R diagrams would have been to reverse engineer the generated schemata. However, the researcher did not have easy

access to a suitable E-R/R based CASE tool and no means could be found of printing or copying the logical diagrams created by *InfoModeler*<sup>TM</sup>. Instead the relevant elements from the *InfoModeler*<sup>TM</sup> logical model were input manually into Visible Analyst. This enabled the creation of an ER/R diagrammatic representation for each view, together with a simple data dictionary. These initial design models are included at Appendix 5.

### 2.2.2 Amalgamation of views

Although INTECoM had allowed for the possibility of leaving the amalgamation of views until the design stage, no guidelines had been provided for undertaking this activity. Therefore, it became necessary to devise a strategy to combine the various ER/R models into an initial global design. It was decided that an examination of apparently similar entities would be an appropriate starting point.

Entity Name	Primary Key	View
Student	Student_id	Student
StudentProgramme	Student_id, Programme_code, Year	Student
StudentPaper	Student_id, Paper_code, Year, Semester	Student
SpecialTopic	Student_id, Paper_code	Student
Research	Student_id, Paper_code, Year	Student
Programme	Programme_code	Student
ProgrammeApproval	Student_id, Programme_code	Student
Paper	Paper_code	Student
Paper	Paper_code	Coord
Student	Student_id	Coord
ResearchInterest	Staff_name, Research_interest	Coord
PaperSemesterYear	Paper_code, Semester, Year	Coord
Enrolment	Student_id, Paper_code, Semester, Year	Coord
StaffAllocation	Paper_code, Semester, Year, Staff_name	Coord
Supervisor	Student_id, Paper_code, Semester, Year, Staff_name	Coord
Student	Student_id	Admin
StudentProgramme	Student_id, Programme_code	Admin
ProposedProgramme	Student_id, Programme, Paper, Year	Admin
CompletedPaper	Student_id, Programme, Paper, Year	Admin
Student	Student_id	HoD
StudentResearch	Student_id, Research_interest	HoD
StaffAllocation	Paper_code, Semester, Year, Staff_name	HoD
StudentQualification	Student_id, Qualification	HoD
StudentEnrolment	Student_id, Programme_code, Paper, Semester, Year	HoD
StudentProgramme	Student_id, Programme_code, Paper, Semester, Year, Staff_name	HoD
ResearchInterest	Staff_name, Research_interest	HoD

**Table 6 Combined list of all entities from analysis model**

While entities that had been given the same name should, theoretically, represent the same objects, it was not sufficient to rely on this heuristic. Instead it was decided to ignore the name of the entity and concentrate instead on an analysis of the composition

of the primary keys of each entity. This was based on the assumption that objects that were identified by the same properties would have a high probability of representing the same thing. If such objects also had attributes in common, it was considered that this would increase the probability. Therefore the first activity was the creation of a table of all 26 entities, their unique identifiers, and the view from which they came (Table 6).

This table was reorganised to allow sorting on the primary key columns. This brought together those entities that appeared to share a common primary key (Table 7). Any differences in the order of the columns in the primary key were not considered significant. With one exception, all entities with duplicate primary keys were then merged and a new entity, containing all the attributes of the originals, was created. The exception concerned the existence of two entities with the same key structure from the same view, ProposedProgramme and CompletedPaper from the College Administrator view. Their names and the fact that they came from the same view, suggested that there might be an important difference between them that should be preserved until the difference could be investigated further. They were thus left as separate entities.

Primary Key	Entity Name	View
Paper_no	Paper	Coord
Paper_code	Paper	Student
Paper_code, Semester, Year	PaperSemesterYear	Coord
Paper_code, Semester, Year, Staff_name	StaffAllocation	Coord
Paper_code, Semester, Year, Staff_name	StaffAllocation	HoD
Programme_code	Programme	Student
Staff_code, Research_interest	ResearchInterest	Coord
Staff_code, Research_interest	ResearchInterest	HoD
Student_id	Student	Student
Student_id	Student	Coord
Student_id	Student	Admin
Student_id	Student	HoD
Student_id, Paper_code	SpecialTopic	Student
Student_id, Paper_code, Semester, Year	Enrolment	Coord
Student_id, Paper_code, Year, Semester	StudentPaper	Student
Student_id, Paper_code, Semester, Year, Staff_name	Supervisor	Coord
Student_id, Programme, Paper, Year	ProposedProgramme	Admin
Student_id, Programme, Paper, Year	CompletedPaper	Admin
Student_id, Programme_code	ProgrammeApproval	Student
Student_id, Programme_code	StudentProgramme	Admin
Student_id, Programme_code, Paper, Semester, Year	StudentEnrolment	HoD
Student_id, Programme_code, Paper, Semester, Year, Staff_name	StudentProgramme	HoD
Student_id, Programme_code, Year	StudentProgramme	Student
Student_id, Qualification	StudentQualification	HoD
Student_id, Paper_code, Year	Research	Student
Student_id, Research_interest	StudentResearch	HoD

**Table 7** List of all entities sorted by their primary key attributes

Overall, this activity resulted in the merger of six entities, Paper, Student, StaffAllocation, Enrolment, ProgrammeApproval and ResearchInterest. As the merger was proceeding, it was noted that other entities had very similar key structures and these were also considered and consolidated as follows.

### 2.2.2.1 Consolidation of Paper entity

The initial composition of this entity from the relevant two views was

<b>Student View</b>	<b>Paper_Coord</b>
*Paper_code	*Paper_no
Points	Title

In theory, these entities should not have been considered at this stage, as the names given to the primary key columns was clearly different. However a close look at the analysis documents, particularly the values given to the attributes in the example sentences suggested that they were synonyms. Paper\_code was adopted as the attribute name and the merged entity was thus,

<b>Paper</b>	*Paper_code
	Title
	Points_value.

### 2.2.2.2 Consolidation of Student entity

The initial composition of this entity from the four views was,

<b>Student View</b>	<b>Paper_Coord</b>	<b>HoD</b>	<b>College Admin</b>
*Student_id	*Student_id	*Student_id	*Student_id
Name	Name	Name	Full_name
Address	Contact_details	Graduate_status	Address
Email_address	Enrolled_college	Position	
Phone_no			

Full\_name was taken as a more detailed version of name, while Contact\_details was taken as a generalised view of the addresses and Phone\_no attributes. The combined entity was thus,

<b>Student</b>	*Student_id
	Full_name
	Address
	Email_address
	Phone_no
	Enrolled_college
	Graduate_status
	Position

### 2.2.2.3 Consolidation of ProgrammeApproval entity

StudentProgramme in the College Administrator view and ProgrammeApproval in the Student view had the same key structure. However the StudentProgramme entity in the student view only differed in the use of Year as a constituent part of the key, and Endorsement as an additional attribute.

College Admin	Student view	Student view
*Student_id	*Student_id	*Student_id
*Programme	*Programme_code	*Programme_code
HoD_approval	Approval	*Year
Recommended_grade		Endorsement

Further investigation revealed that the Endorsement was not specific to the year of a programme in that only one endorsement of a programme was of interest. If the endorsement was changed at any time, only the current information was required. It was concluded that the Year element was of interest to the students only because they needed to renew their enrolment in the programme on an annual basis. The three entities were thus combined without the Year attribute as follows,

ProgrammeApproval	
	*Student_id
	*Programme_code
	Endorsement
	HoD_approval
	Recommended_grade

### 2.2.2.4 Consolidation of StaffAllocation entity

There were two StaffAllocation entities with the same primary key and no attributes.

Paper Coord View	HoD view
*Paper_no	*Paper_no
*Semester	*Semester
*Year	*Year
*Staff_name	*Staff_name

These were merged with no alteration into the new entity StaffAllocation.

### 2.2.2.5 Consolidation of Enrolment entity

Each view contained one or more entities that appeared to reflect the student's enrolment in papers, although they had been named differently and had minor differences in the primary key. These were StudentPaper from Student view, Enrolment from the Paper Co-ordinator view, StudentEnrolment from the Head of Department View and ProposedProgramme and CompletedPaper from the College Administrator view. The primary difference was that neither the student nor the paper co-ordinator view had recognised the enrolments as being dependent on the programme as this was implicitly understood. It was considered that adding the Programme\_code to the primary key of the entities of these two views would not have a significant effect on the views. However the ProposedProgramme and CompletedPaper, which omitted Semester from the primary key, were clearly interested only in the year in which papers were (to be) taken. Consequently these two entities were not considered suitable for amalgamation at this point and were put aside for later consideration.

<b>Student view</b>	<b>Paper Coord</b>	<b>HoD</b>
*Student_id	*Student_id	*Student_id
*Paper_code	*Paper_code	*Programme_code
*Year	*Semester	*Paper_code
*Semester	*Year	*Semester
	Approved_content	*Year
	Result	Approved_content
	Project_title	Result
		Project_title

However, the remaining three entities were merged to create a new entity Enrolment, identical to the entity StudentEnrolment in the Head of Departments view.

### 2.2.2.6 Consolidation of ResearchInterest entity.

The entities named ResearchInterest in the Paper Co-ordinator view and the Head of Department view were identical. The new entity was renamed StaffResearch.

18 entities had now been identified and these are shown at Table 8. Each of these entities were examined and compared against the other entities. Similarities in key structures and possible synonyms were specifically targeted to see if they provided an opportunity to make further mergers. Three entities in particular seemed worth further analysis, PaperSemesterYear, SpecialTopic and Research.

Primary Key	Entity Name	View
Paper_code	Paper	Coord, Student
Paper_code, Semester, Year	PaperSemesterYear	Coord
Paper_code, Semester, Year, Staff_name	StaffAllocation	Coord, HoD
Programme_code	Programme	Student
Staff_code, Research_interest	ResearchInterest	Coord, HoD
Student_id	Student	All
Student_id, Paper_code	SpecialTopic	Student
Student_id, Paper_code, Semester, Year	Enrolment	Coord, Student
Student_id, Paper_code, Semester, Year, Staff_name	Supervisor	Coord
Student_id, Programme, Paper, Year	ProposedProgramme	Admin
Student_id, Programme, Paper, Year	CompletedPaper	Admin
Student_id, Programme_code	ProgrammeApproval	Student, Admin
Student_id, Programme_code, Paper, Semester, Year	StudentEnrolment	HoD
Student_id, Programme_code, Paper, Semester, Year, Staff_name	StudentProgramme	HoD
Student_id, Programme_code, Year	StudentProgramme	Student
Student_id, Qualification	StudentQualification	HoD
Student_id, Paper_code, Year	Research	Student
Student_id, Research_interest	StudentResearch	HoD, Student

**Table 8 List of entities after initial merging**

### 2.2.2.7 Consideration of potential mergers

**PaperSemesterYear** A merger was considered with StaffAllocation, as the key structures were similar. However this entity had two attributes, Points\_value and Type neither of which required Staff\_name to identify them, hence the entity was left as it was.

**SpecialTopic** - A special topic had been identified as a subtype of Paper and appeared to relate to the Type attribute of the PaperSemesterYear entity. It also appeared that its attributes Content\_approval and Content were directly related to the Approved\_content attribute of the Enrolment entity. Although it did not include the semester and year attributes as part of the primary key, it was determined that in fact they were required to uniquely identify these attributes. Consequently this entity was removed but a note added to the Enrolment entity that if the Type attribute of PaperTypeYear was set to a value of 'Special Topic' then a value of Approved\_content was required.

**Research** - A research paper has also been identified as a subtype of Paper. It looked as though it too could be subsumed by Enrolment. It was also clear that the attribute Title referred to the title of the research project undertaken by a student. This had been omitted from the student view. Consequently this entity was removed but a note added to the Enrolment entity that if the Type attribute of PaperSemesterYear was set to a

value of 'Research' then a value of Title was required. An additional rule was also added that if the value of Type was 'Research' then there must be at least one matching entry in the Supervisor entity.

At the end of this process 14 entities had been identified and an entity/attribute list is shown at Appendix 5.5.

### **2.3 Generating and evaluating alternative solutions**

This final draft design model served as the beginning point for the design of the actual data structures that could be implemented in a relational database. Once this model was scrutinised a number of 'improvements' were incorporated.

The first area of concern was the two entities with identical primary keys, i.e. ProposedProgramme and CompletedPaper. It was also noted that apart from the missing Semester attribute and the use of the Programme name rather Programme\_code these entities had the same key as Enrolment. It was determined that these entities were potentially derivable and the decision was made to delete both of these entities. Examining these entities involved looking at the relationship between the Supervisor and the Staff\_name, which the developer assumed to be synonyms. It was also noted that Staff\_name was used elsewhere in the model but was not a foreign key. The creation of a Staff entity would allow the use of a Staff\_code attribute and provide referential integrity.

While considering these entities, it was also decided to place the College name and the name and telephone number of the College Postgraduate Administrator as attributes of Programme, thus providing information on which person in which college needed to be notified of both the proposed and completed programmes.

The second area for scrutiny was the use of the Qualification attribute as a part of the primary key of StudentQualification. In order to utilise this attribute effectively, it seemed necessary to create a qualification entity and use a qualification code as a primary and foreign key. This would allow referential integrity and would provide consistent entry of qualifications. After investigating the use of this attribute it was decided that it was required only as occasional background information by the Head of Department and could be adequately stored as a textual comment. The entity StudentQualification was thus deleted and a qualification attribute was added to the

Student entity. Following this decision it was decided that the previous work experience of the student might also be useful and was accommodated in a similar way.

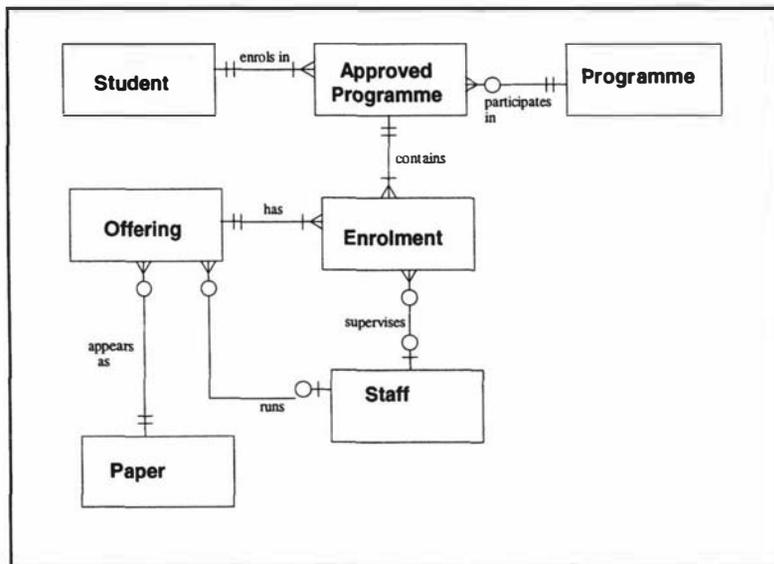
The third area for scrutiny was the use of the `Research_interest` attribute as part of the primary key of two entities, `StaffResearch` and `StudentResearch`. The idea of having a coded list of research interests was considered useful but beyond the scope of this development. However, the initial indication of research interests of students intending to enrol in postgraduate study could be easily captured and could also be usefully held as a textual comment and likewise the general areas of the research projects that were offered by staff. Consequently the entities `StaffResearch` and `StudentResearch` were deleted and a `Research_interest` attribute added to both the `Student` and the `Staff` entities.

The fourth area concerned the recording of more than one supervisor for some research projects. It was decided that it was only necessary to store an indication of the chief supervisor. An optional relationship from `Staff` to `Enrolment` was added thus allowing for the recording of a `Staff_code` for those papers that have a `Type` of 'Research'. In this way, both the key-only entities `StudentProgramme` and `Supervisor` could be deleted with no loss of additional information.

The issue of the `Year` attribute, originally part of the primary key for the `ProgrammeApproval` entity was re-visited. It was definitely not needed as part of the key, however it would be useful to know in which year approval was first given for a programme. The attribute `Start_year` was thus added to this entity. Additionally the entity was renamed `ApprovedProgramme` to reflect the fact that this was also the programme that the student could be expected to complete.

`Staff Allocation` was now the only entity with no attributes other than its primary key. While acceptable, it seemed worth investigating particularly as it inherited 3 parts of its key from `PaperSemesterYear`. As only one staff member was ever recorded as offering a postgraduate paper there seemed no reason why the `StaffAllocation` entity could not be replaced with a relationship from `Staff` to `PaperSemesterYear`. At the same time `PaperSemesterYear` was renamed `Offering`.

The number of entities had now been reduced to seven and a new E-R diagram, termed the second draft design model and reflecting all these changes, is illustrated at Appendix 6.1 and a diagrammatic representation is shown at Figure 19.



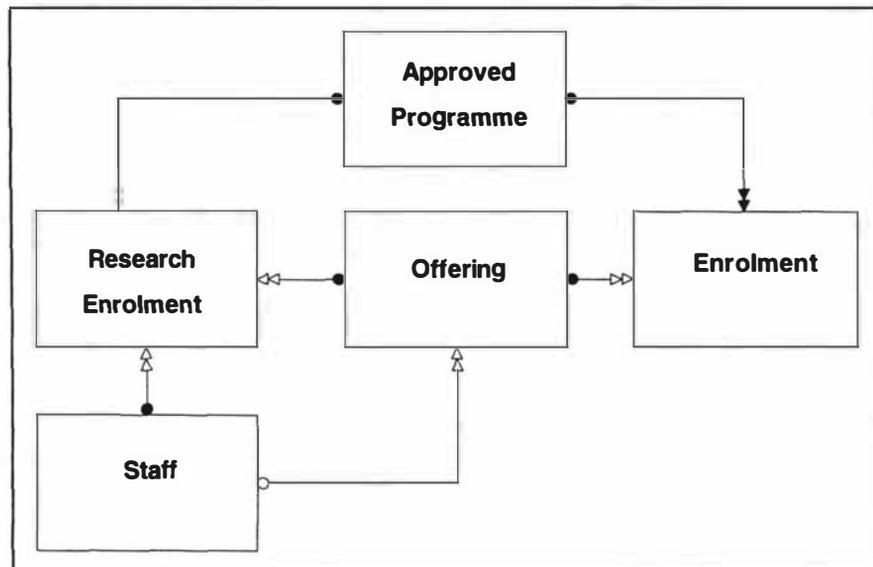
**Figure 19 Second draft design model**

The general design represented in this second draft appears to be much simpler and 'cleaner' than the initial design model. It has been adapted to take advantage of relational data structures and is arguably easier to comprehend than its predecessor. The concentration of attributes in the Student and Enrolment entities would seem to clearly reflect the intended scope and emphasis that the development is intended to build. One area of obvious concern remained: the two optional relationships from Staff to Offering and Enrolment which seemed to provide a 'non-elegant' solution and which could benefit from further work.

Several alternatives were considered for these relationships. Four possible solutions seemed worth pursuing.

a) Create a mandatory relationship between Staff and Offering (Relationship 1) and remove that between Staff and Enrolment (Relationship 2). It was determined that the Head of Department took overall responsibility for all research papers and could therefore be entered as the paper co-ordinator. However there were some negative implications of this solution. The staff allocation information could not now be completely satisfied, as the Offering relation would not provide complete information. It would also not be possible to satisfy the College Administrator's need to know the name of the Supervisor of a research project.

b) Make Relationship 2 mandatory and remove Relationship 1. This solution had none of the disadvantages of the previous one. The complete staff allocation information was available via the Enrolment relation and this included the research supervision of staff members. This would be achieved at the expense of unnecessarily duplication of the Staff\_code; i.e. the Staff\_code would now be recorded for every individual student enrolment, rather than just once for the offering. Initially this seemed an acceptable solution but a more significant problem was identified in that no staff member could be shown as responsible for a paper unless students were enrolled in it. This restricted the ability of the database to store future intentions, e.g. to record who would be co-ordinating a paper in the second semester.



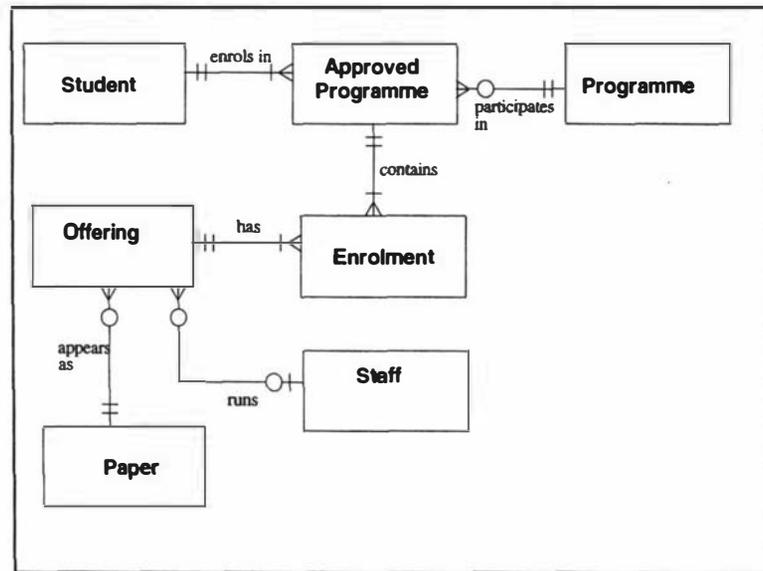
**Figure 20 One solution to the Research Supervisor Problem**

c) A third solution, illustrated at Figure 20 seemed to be to split the Offering relation into two, one for research offerings and one for all the rest. Thus Relationship 1 would become mandatory in that all offerings would require a staff member, while a similar relationship would be created between Staff and the new relation ResearchEnrolment. Two additional relationships were then required, one between ApprovedProgramme and ResearchEnrolment and another between Offering and ResearchEnrolment. The primary key of the new relation would consist of Student\_id, Programme\_code, Paper\_code, Year and Semester, identical to that of

Enrolment. The new relation would also require the attribute Result, and would take the attribute Project\_title from Enrolment.

This solution is interesting for a number of reasons. First of all it does little to alleviate the original concern. A fully optional relationship still exists between Staff and Offering and there is still an additional relationship emanating from Staff. In addition a new relation had been created and this necessitated another new relationship between ApprovedProgramme and ResearchEnrolment. Despite the additional complexity of this solution, it seemed, to the designer, to offer a more satisfactory and semantically richer structure. Nevertheless, a number of the functional requirements would become clumsier to accommodate. The requirement to identify the staff allocation could be fulfilled but only by querying two relations as could the College Administrator's requirement to know the name of the supervising staff member. Neither of these problems was significant in themselves. However, the original design had provided for all the information about which papers a student was enrolling in for a particular programme, to be retrieved from one relation. The new solution would always require both relations to be queried as there was now no way of knowing whether or not a student was enrolled in all research papers, all non-research papers or a mixture of the two. While again not an insurmountable problem as such, it contributed to a feeling of unease with the proposed structures.

These three solutions were re-visited and the first and second were dismissed as creating more problems than they solved. The third solution appeared to introduce complexity into the structure for little gain and it seemed that the original structure should be retained. However further thought prompted a much simpler alternative which appeared to remove the objections to the original structure, support all the requirements (although the staff allocation still required the querying of two relations) and could be achieved at only a small cost.



**Figure 21 Final draft design model**

This fourth solution entailed removing the relationship between Staff and Enrolment, and returning the attribute Supervisor to the Enrolment relation. The implications of this solution were that referential integrity was lost with the removal of the Staff\_code foreign key from Enrolment. On the other hand this also provided some flexibility in allowing the recording of a supervisor from outside the department who would not normally be recorded in the Staff relation. While this was considered unlikely, a situation requiring precisely this flexibility occurred as the solution was being developed. It was thus decided to adopt this solution and thus reverse one of the untimely design decisions made earlier.

This completed the first major phase of the design activities and the model created at this point was termed the final draft design model (Appendix 6.2) and a diagrammatic representation is shown at Figure 21.

## **2.4. Create and verify the final design model.**

### **2.4.1 Syntactical check**

The final draft model was now checked by the designer for syntactic and structural errors. The first activity was to check that the primary key to foreign key relationship links shown on the diagram made sense in terms of the relation descriptions and

represented appropriate cardinality and optionality. A set of two-way sentences, illustrated at Table 9 was created from the diagram to assist in this activity.

1	Each ApprovedProgramme	must be enrolled in by only one	Student
2	Each ApprovedProgramme	must belong to only one	Programme
3	Each ApprovedProgramme	must contain one or more	Enrolment(s)
4	Each Enrolment	must belong to only one	ApprovedProgramme
5	Each Enrolment	must belong to only one	Offering
6	Each Offering	must have one or more	Enrolment(s)
7	Each Offering	must belong to only one	Paper
8	Each Offering	may be run by only one	Staff
9	Each Paper	may appear as one or more	Offering(s)
10	Each Programme	may participate in one or more	ApprovedProgramme(s)
11	Each Staff	may run one or more	Offering(s)
12	Each Student	must enrol in one or more	ApprovedProgramme(s)

**Table 9 Two-way sentences for the final draft design model**

One error was corrected during the creation of the two-way sentences where the diagram had shown a fully mandatory relationship between Offering and Enrolment. Clearly it was possible for an offering to be available but for no enrolments to have been received. The mandatory nature of the relationship between Student and ApprovedProgramme was also questioned and it was decided that a student who was not enrolled in a programme would not be entered on the system. The relationships had only been named in one direction on the diagram. These names were used where possible but were adapted or invented as necessary. There was no attempt to utilise the language of the analysis documents.

No errors were detected in this phase and it was noted that with the exception of Staff\_code in the Offering relation all other foreign keys also participated in the primary key of their host relation. A complete list of these links is shown at Table 10.

Relation 1	Primary Key	Relation 2	Foreign Key
ApprovedProgramme	Student_id, Programme_code	Enrolment	Student_id, Programme_code
Offering	Paper_code, Semester, Year	Enrolment	Paper_code, Semester, Year
Paper	Paper_code	Offering	Paper_code
Programme	Programme_code	ApprovedProgramme	Programme_code
Staff	Staff_code	Offering	Staff_code
Student	Student_id	ApprovedProgramme	Student_id

**Table 10 Primary to foreign key links in final draft design model**

### 2.4.2 Check Normalisation

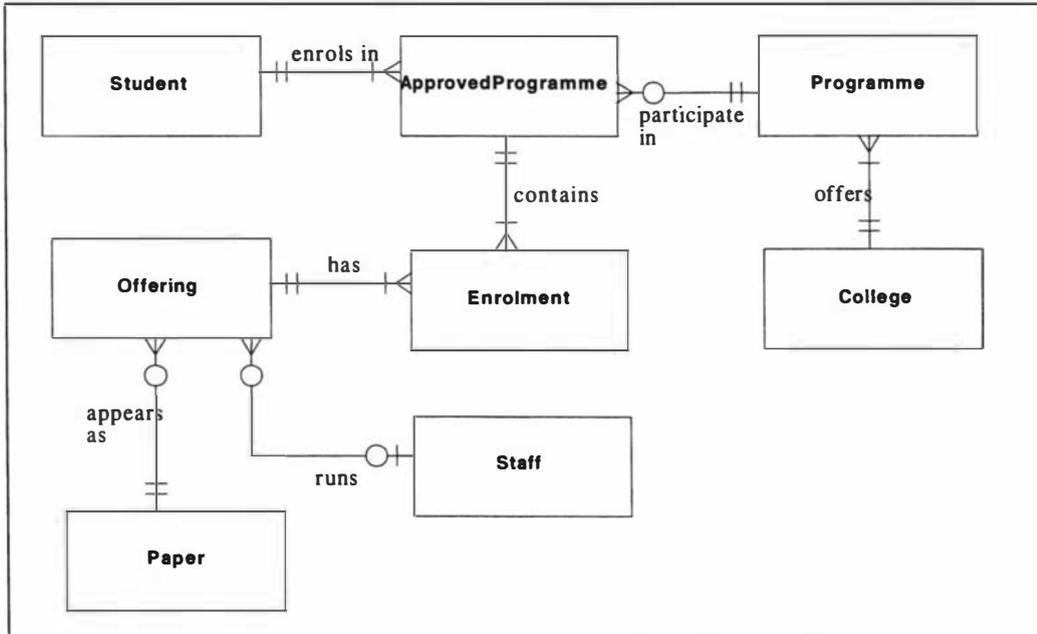
A check was then made for normal form conformity. It was decided to restrict this check to Boyce Codd Normal Form. Scrutiny of the attributes detected three errors. The Type attribute of the Offering table was not functionally dependent on the whole key, as it did not require the semester and year. It was thus moved to the Paper entity. It was also recognised that the Points\_value attribute did not require the Semester attribute to identify it. To conform to 2NF it thus needed to be removed to a separate relation, which should hold Paper\_code and Year as the key with Points\_value as the only attribute, thus leaving the Offering relation as a (required) key-only relation. While recognising the correctness of this approach it was considered trivial and a certain candidate for denormalisation in the physical design phase. It was thus left unchanged. The Points attribute in the Paper relation was also in error; it was recognised as being the same attribute as Points\_value and thus needed Year to identify it. It was removed from the Paper relation.

This investigation of the points attribute also encompassed the Points\_value of the Programme relation. It was initially thought to be derivable, as the total number of points undertaken by a student within a programme. However, further investigation, again using the examples as a useful starting point, identified it as being the minimum number of points required for a student to complete a program. It was therefore renamed as Required\_points. The relation Programme was also not in BCNF as the attributes PG\_admin\_name and PG\_admin\_phone\_no were functionally dependent on College that was not a candidate key of the relation. A new relation, College, with College\_name as the primary key and these two attributes was created.

As a result of this investigation it was also realised that the Student relation contained an attribute Enrolled\_college. This attribute was considered redundant, as the information was derivable via the ApprovedProgramme relation. It was thus removed. It was noted that the attributes, Previous\_qualifications, Work\_experience and Research\_interest were not likely to be logically atomic. However, following the previous discussion on these issues, these attributes were left unchanged.

This completed the first round of verification and another version of the model, termed the Pre-verification Design Model, was constructed and is included at Appendix 6.3.

The diagrammatic representation is also shown at Figure 22. This model was then used as the basis of the semantic verification.



**Figure 22 Pre-Verification Design Model**

### 2.4.3 Initial Semantic Verification

#### 2.4.3.1 Create NaLER Sentences

In order to complete the semantic verification of the model a complete set of NaLER sentences was now created according to the steps described in Chapter 10. The steps were applied to the design model as follows.

##### *a) Document the model conventions*

The model conventions that had been used were documented as,

- entities were shown as rectangles enclosing their name,
- relationships were shown by connecting lines for the primary/foreign key links,
- cardinality of relationships was shown by a circle at the 'one' end and a double headed arrow at the 'many' end,
- optionality was indicated by a clear circle or arrowhead,
- mandatory participation was shown by black circles or arrowheads,
- attributes participating in a primary key were designated by an asterisk,
- foreign key columns were italicised.

*b) Check assumptions*

All the assumptions were worked through, thus providing an additional syntactical check. It was noted in line with Assumption 6 that both ApprovedProgramme and Enrolment were entities that were resolving many to many relationships between 2 entities. Additionally it was noted that relationships had only been named in one direction. However, the construction of the two-way sentences had created an *ad hoc* complete set of such names which would be used initially but which would be adapted when appropriate.

*c). Simple entities*

*Construct primary key sentences*

Each entity had a well-defined primary key and no difficulties were experienced in constructing these sentences.

*Construct attribute sentences*

No problems were encountered in this phase.

*Construct relationship sentences*

No problems were encountered in this phase. A complete list of the sentences created in these 3 stages is shown at Appendix 7.1.

*Construct super/subtype sentences*

There were no super/subtype constructs.

*d) Complex entities*

There were no problems with the complex entities.

*e) Populate with examples.*

In accordance with the NaLER guidelines examples were not constructed for the primary key sentences and examples were taken primarily from the original analysis documents. As it was necessary to work from these documents it was straightforward to record the view or views and the number of the fact type sentences from which the examples were being taken and note any differences. If an indirect translation was required from the original fact this was recorded as 'related to' the fact and one apparent conflict was also noted. This proved to be extremely useful in later stages. The full list of the NaLER example sentences is included at Appendix 7.2.

*f) Produce NaLER description.*

The complete set of sentences and examples together make up the ‘NaLER description’ and therefore, no further work was required.

#### **2.4.4 Create Equivalence Table**

The arguments for INTECoM had suggested that the NaLER sentences could be used for verification by the users but had not described a method for doing so. It was decided to draw up a table to correlate these sentences and the original fact types of the analysis model. This activity was made significantly easier by the cross-referencing that had already been done during the creation of the example sentences. In addition it provided a useful opportunity for the developer to check that all the user requirements had been considered. It was hoped that this should minimise any user dissatisfaction and ensured that the developer would be familiar with any differences between the original user views and their final relational form.

During the creation of this correlation table, various issues were noted as requiring further investigation. One straightforward error of omission was discovered in the model where the Endorsement attribute had been inadvertently lost from the ApprovedProgramme relation. Additionally sentences had not been created for the Required\_points in the Programme relation and the Research\_interest in the Staff relation. These mistakes were rectified and the relevant example sentences created. Three facts, all from the Head of Department’s view did not appear to have been supported in the final design and these were analysed more closely.

*F14 StudentProgramme is for TimePeriod.* Judging by the examples that had been provided this fact appeared to be recording the time period in which a student was expecting to complete a Programme. In theory, it would be derivable from the student enrolments but only if the intended future enrolments were entered as well as the actual current enrolments. The developer was aware that future intentions were sometimes recorded but that many students did not have a specific plan for future years. Further discussion with the user made it clear that on occasion, particularly when planning programmes for international students, it was desirable to know when a student intended to graduate. Consequently, the developer added the attribute Intended\_finish\_year, to the ApprovedProgramme relation to satisfy this requirement.

*F16 StudentProgramme requires College\_approval.* The examples provided for this fact to record a simple 'yes' or 'no' to show that final approval had been given to the student's enrolment in the programme by the Graduate Studies Committee of the relevant College. It was considered important to record this information although there was some concern as to how accurately it could be collected. In addition, the recording of the date on which College approval was granted was thought to be more useful and less prone to error, than a simple 'Y' value. As a result the attribute *College\_approval\_date* was added to the *ApprovedProgramme* relation. It was also noted that this attribute would need to be optional as not all programmes received explicit approval from the Colleges.

*F17 StudentProgramme requires Department\_approval.* Although this attribute appeared to hold a different set of values from the *HoD\_approval*, which had come from both the Student and the College Administrator's view, it was clear that it was intended to record similar information, namely that HoD approval had been granted. The primary difference was seen as being one of scope in that within the proposed system there was only an interest in recording whether the IS Head of Department had given approval to the programme. There was no requirement for the system to establish whether students from non-IS programmes who were enrolling in an IS postgraduate paper, had the approval of their own Head of Department. It was therefore considered sufficient to equate this fact with the existing sentence S9. However it was decided to be consistent with the previous decision and record this information as the date on which approval was granted. The attribute was thus renamed *HoD\_approval\_date* and recorded as optional. A new sentence was then created to replace the original.

One final addition was made to the model. At the point at which the development had begun, postgraduate papers in the IS department had only been offered to internal students. However, during the course of the design phase the department made a decision to offer some papers in 'block' mode for the following academic year. The developer thus considered it might be prudent to record the mode in which a paper was being offered and another new sentence was created to accommodate this.

## **2.5 User Semantic Verification**

In order to facilitate the user verification of the design model, the NaLER sentences were assembled into groups relevant to the original views. The correlation table was

used to facilitate the creation of the tables shown in Appendix 8. These 'equivalence' tables, one for each view, hold the number of the original fact type, the number of the corresponding NaLER sentence and the NaLER sentence itself. A set of verification documentation was then assembled for each view. Each set consisted of the relevant equivalence table, the original fact types and examples for the appropriate view and, the list of all NaLER example sentences.

A representative of each user group, except the College Administrator was then presented with this documentation. The purpose of the exercise, and of each of the documents was also explained. Verification consisted of the developer talking through each fact type and corresponding NaLER sentence. Where the connection was tenuous or unclear, the various examples were used to provide confirmation that the need was still being met. At no time were the users shown either the relational schema or the ER/R diagrams, although all the users would have been capable of understanding them.

## **2.5 The Outputs**

INTECoM envisages that the datalogical output of this phase, the design model, consists of a standard ER/R deliverable. Two of the three elements of this are included at Appendix 9. A full data dictionary description was not maintained for this example but much of the relevant information is to be found within the relational schema. The infological model is regarded as being the set of NaLER example sentences derived from the data structures. In addition, the NaLER sentences were used to create a new ORM, which, in turn, required a new set of fact types, both of which are included at Appendix 10. The fact types can be viewed as another aspect of the infological version of the design model. To ensure their accuracy and as a final check for completeness, a table was generated cross-referencing these final fact types to the NaLER sentences (Appendix 11). These fact types were also used to generate a new logical model and set of related SQL statements from *InfoModeler*<sup>TM</sup> as a final check that the design model was syntactically correct. The model generated by *InfoModeler*<sup>TM</sup> was identical to that created by the design process.

## **3. Create the Physical Model**

For the purposes of testing the process outlined in the INTECoM framework, the final two steps of creating the physical model and the physical user views were not

considered relevant. Therefore no further discussion is included here. However, the design model was fully implemented with only minor modifications necessitated by physical considerations.

## Observations

Following the working of this example, a number of observations can be made, some related to the framework itself and others to issues that have already been raised in the course of developing and justifying INTECoM.

*The collection of examples.* The opinion had been voiced in informal discussions with colleagues that the collection of a complete set of examples would not be possible and that this inability would reflect adversely on the outcome of the analysis. Indeed, neither NIAM nor INTECoM can guarantee that a complete list of examples has been gathered and this could impact on the specification of constraints and result in the non-identification of roles. However, in practice this did not appear to be a significant problem. In general, the request to provide examples resulted in a very full set and initially did lead to the identification of new fact-types. The disciplined behaviour required to collect and document a full set of examples may actually be seen as an improvement on the more *ad hoc* methods used in the E-R tradition. No problems were identified in the ISPG development that stemmed from an inability to collect sufficient examples. While some fact types were added during the design phase they all resulted from legitimate design activities and not from a deficiency in the analysis.

*The use of examples.* The possession of a detailed set of examples was extremely useful in several situations. During the amalgamation of views, it became important to identify synonyms and homonyms and reference to the values within the relevant example sentences provided the developer with a sound basis for decision making in these areas. For example, it required little effort to identify without doubt that *Paper\_code* and *Paper\_no* were synonyms. While perhaps a trivial example, there is a clear potential for identifying significantly less obvious connections. This technique was also used in assisting in the identification of those relations that could be merged.

In addition, the existence of example sentences allowed the developer to ensure a significant amount of semantic veracity as the development progressed. At both, the end of the analysis phase, and particularly, at the point of creating the NaLER sentences,

the existence of the initial example sentences was invaluable. In the latter case it provided the developer with an invaluable tool to identify whether all the analysis 'facts' had been represented in an appropriate form. The users too, were enthusiastic in their use of the examples. Although they were not asked to specifically comment on the examples, observation during the user verification activities identified that on several occasions users utilised the examples as an aid to understanding.

*The use of analysis views.* The original definition of INTECoM had suggested that the amalgamation of the initial user views could either be undertaken as the analysis progressed or as the first step in the design phase. The ISPG development clearly demonstrated the benefits of preserving each unique user view throughout the analysis phase. Firstly, as the eventual amalgamation activity showed, the process of amalgamation requires decisions of a design nature to be made and that in areas of conflict, some views should be afforded a higher priority than others. A major element in the justification for the development of INTECoM rested on the principle that design decisions should be avoided in the analysis stage wherever possible. In line with this principle and these observations, INTECoM should be revised to recognise that the amalgamation of the user views should occur as the first step in the design phase. In addition to these observations, it was also noted that although completeness cannot be guaranteed, nevertheless, retaining all aspects of all views throughout the analysis stage minimises the loss of essential facts. In addition, it seem significantly safer to consciously choose to omit a fact during the design phase than to decide, perhaps early on in analysis, that a fact is either not important or can be accurately captured by an alternative construct in a different user's view.

*The amalgamation of user views.* INTECoM does not provide any guidelines on how this amalgamation could be achieved and it thus became necessary to develop a feasible strategy while the development was underway. The steps within the amalgamation phase can be summarised as follows

- Create an initial E-R/R model for each independent view.
- Create one combined list of all entities and their primary keys, from all the views.
- Sort list on the names of the primary key attributes and identify those entities with apparently identical primary keys.

- Use the examples to confirm which entities were essentially representing the same object.
- Merge entities as appropriate.
- Create revised list of entities and critically examine to identify synonyms among the entity names and primary key attributes and merge as appropriate.

In the example, this process alone reduced the number of entities from 26 to 14. However, more complex decisions were required for the remaining stages of design. It became clear that the merger process required skills, based on previous experience and these could not be definitively prescribed but only generally described. This tended to support the previous arguments that much of the work of the design stage would be reliant on the experiential skills of the designer.

*The need for auditability.* The instantiation of the NaLER sentences, with examples taken from the original analysis model, provided an obvious and natural opportunity to identify which NaLER sentences corresponded to which analysis fact types. As the examples were created, a close record was kept to document this correspondence and this resulted in the 'equivalence' tables shown in Appendix 8. The construction of this table produced significant benefits in the later, verification stages of the design phase as it was used,

- to check that all fact types identified in the analysis has been appropriately incorporated into the design model, and where this was not the case, allowing remedial action to be taken before the presentation of the model to the users,
- to re-create the appropriate user views with the NaLER sentences, thus eliminating the need for each user to comprehend the entire design model, and
- to ensure that the developer could account for the all the 'facts' contained in the design model, tracing their individual existence back to either a user requirement, the recognition of a future or functional requirement or the constraints of the database paradigm.

Recognition of the importance of these three functions led to the addition of 'auditability' as a quality goal and this is discussed together with other quality considerations, in Chapter 12. It was also noted that the existence of this audit trail

provided an (unquantified) high level of confidence in both user and developer. The developer had had to justify the connection between a certain analysis fact type and its equivalent NaLER sentence and was very familiar with the situations where the correspondence was not exact. The developer was thus able to predict likely questions and prepare clear explanations. In addition, the users could quickly verify the 'facts' that were clearly the same in both models and concentrate instead on those which appeared to have changed. Both the developer and the users felt that this reduced the time necessary for verification and increased their confidence that the verification was accurate.

## Summary

A database, which has been subsequently implemented and used, was successfully designed using INTECoM. The framework was found to be useful, useable and required little refinement. The exercise was not intended to be and cannot claim to be an effective evaluation of the framework. The researcher, as developer, obviously had a vested interest in the success of the exercise, the users were all experienced developers themselves and the development itself was not large enough to generate the level of complexity with which INTECoM is designed to assist. It was very difficult to extract true 'user views', as the users were too knowledgeable and adept at second guessing what the developer needed to hear. Nonetheless, in all communication with users every attempt was made to avoid reference to the final database and during the final verification steps, none of the users asked to see the E-R diagram, seeming content to accept the NaLER sentences as a true representation of the data structures. In addition, the developer being an experienced 'traditional' data modeller had difficulty in avoiding 'relational thinking' during the analysis stage.

Nevertheless, the working of this example was a valuable means of providing an initial test of the theory represented by the INTECoM framework. Apart from the additional techniques that needed to be developed, there was no point at which the framework was altered or not followed in the form in which it was originally described in Chapter 9. The general principles behind the need to distinguish between the analysis and design phases of a database design exercise and the benefits of matching appropriate tools to appropriate activities were confirmed.

# 12 INTECoM: quality matters

*"The proof of the pudding is in the eating" Proverb*

## **Introduction**

Although the literature examined in Chapter 8 did not provide a definitive set of quality goals and measurements, it did highlight the importance of attempting to assess the 'quality' of the final product of a modelling exercise. However, none of the literature considered assessing the quality of the modelling process itself nor measuring the 'fitness of purpose' of various modelling techniques to the activities that they were being used to undertake. Likewise, the quality plan initially defined for the ISPG INTECoM development focussed exclusively on the quality of the final artefacts. However, as the development proceeded, it became clear that this initial plan was not adequate. Several aspects of the development would not be assessed and several inherent strengths of INTECoM would not be fully realised. This chapter begins by describing the initial quality plan and then discusses three significant issues that became apparent as the plan was put into practice. As a result of these observations, the plan was abandoned and a far more substantial quality framework, firmly rooted in both the research literature and the theoretical considerations underpinning INTECoM, was developed and used. This new quality framework and its instantiation are also detailed in this chapter, as are a number of observations on the use of both INTECoM and the quality framework. An interesting consequence of this focus on quality assessment was the addition of certain refinements to INTECoM, which resulted in some minor modifications to the inputs of the first two stages. An amended diagram of INTECoM is illustrated on page 206.

## **Initial Quality Plan**

Based on the work of Krogstie *et al.* (1995) and Moody and Shanks (1994), an initial set of quality criteria was determined for assessing the final products of the INTECoM development. An overview of these criteria, two for the infological analysis model and three for the datalogical design model, is shown at Table 11.

Model	Quality	Quality Goal	Means
Analysis	<i>Perceived Semantic</i>	Semantic correctness a) perceived validity b) perceived completeness	Subjective ratings that all the statements made in the infological model are correct and relevant, and that all such statements are included.
Analysis	<i>Pragmatic</i>	Pragmatic correctness a) understandable b) understood	Subjective ratings that the infological model is understandable and has been understood.
Design	<i>Syntactic</i>	Syntactic correctness	Syntax check of the datalogical design model.
Design	<i>Social</i>	Social agreement	Subjective rating by the designer that any apparent conflict between users of the model are based in their individual perception of the model rather than in its structure.
Design	<i>Structural</i>	a) Simple  b) Flexible	That the datalogical model contains the minimum number of entities and attributes necessary to represent the required statements.  That the datalogical model is at least in BCNF

**Table 11 Initial quality criteria for INTECoM design model**

These criteria combine four of the qualities described by Krogstie *et al.* (1995), (*perceived semantic, pragmatic, syntactic and social*), with two goals identified by Moody and Shanks (1994), (*simplicity and flexibility*). Krogstie *et al.*'s 'qualities' encompass all the identified requirements of an infological model and are represented by the first four quality criteria in Table 11. The only aspect not included is the 'semantic quality', which, as Krogstie *et al.* (1995) themselves observe, is an ideal, impossible to establish or check directly. Instead the more relevant and accessible, 'perceived semantic quality' is substituted.

A reconsideration of the other literature discussed in Chapter 8 suggests that a model should also be assessed on its fitness for its intended purpose and clearly some of the quality goals summarised in Table 5, relate to the structural aspects of the model itself. The quality goals not covered by Krogstie *et al.*'s (1995) framework include simplicity, flexibility, integration, implementability, maintainability, suitability and performance. Of these, the first two appear both relevant and measurable and were thus included. The rest were discarded for the various reasons described below.

Moody and Shanks have defined integration as the measure of how well a model fits with other organisational models. While this is recognised as an important quality in some contexts, it was not relevant in the 'stand alone' environment that was intended for the ISPG development, although it should be included for other developments where

appropriate. Implementability (Moody & Shanks, 1994) and maintainability and performance (Kesh, 1995) are more closely related to the characteristics of the final physical database schema than to either the infological or datalogical conceptual data model and were therefore not included. Finally, suitability (Kesh, 1994), while clearly intended to address the overall fitness of the data model for its purpose, relied only on rather vague subjective ratings. No alternative means of measurement suitable for the intended development could be found and therefore it too was discarded.

The infological version of the analysis model would provide the basis for judging two components, semantic and pragmatic quality. *Semantic quality* requires that all statements contained in the model are correct and relevant and that all such statements are included. It becomes a user responsibility to confirm that no statements (fact types) are missing, that all the statements represent valid information and that the example sentences for each fact type are correct. *Pragmatic quality*, i.e. that all the statements are understandable and understood, is thus a necessary pre-requisite for this check and in practice will need to be achieved before any semantic verification can begin.

The datalogical version of the design model would provide the basis for assessing the remaining quality goals. *Social quality* requires the designer to confirm that the model can support all the required user views and that any apparent conflicts in these views are a result of differing interpretations of the model rather than any incompatibility inherent in its actual structure. A subjective rating by the designer, and possibly from a peer review would be used to measure this. As the datalogical design model for the ISPG system was to be an ER/R hybrid, *syntactic quality* would require that the model be relationally correct and this would be assessed by checking the model for relational conformance<sup>1</sup>. Again, this would be the responsibility of the designer assisted by peer review. *Structural quality* has two distinct goals, flexibility and simplicity. Unlike Moody and Shanks (1994), who measured flexibility subjectively by its perceived ability to support changes in requirements, flexibility would be measured here by assessing the level of normalisation of the model. This decision was made on the basis that normalisation will generally provide a “good base for future growth” (Date, 1995,

---

<sup>1</sup> This rather narrow definition proved constraining. The commonly used generalisation-specialisation construct is not strictly correct from a relational viewpoint and is yet a common and useful approach in modelling at this level.

p.335) and Boyce Codd Normal Form is taken as being the minimum level of acceptable normalisation. Following Moody and Shanks (1994), the simplicity of the model would be measured by calculating the number of entities and relationships in the model with the aim of achieving the smallest number that would support all the required statements.

### **The Quality Plan in Practice**

Thus, the ISPG development began with a set of quality criteria developed from the relevant literature, which, in line with much professional practice, would be applied at the end of the process. However as the development was under way a number of quality related issues became apparent. Three of these issues were sufficiently significant to require a substantial review of the quality plan and are discussed below.

#### ***1. Analysis or Design Model?***

The quality criteria discussed in Chapter 8 are primarily concerned with evaluating the final product of a conceptual modelling process but just as no clear differentiation is made between analysis and design models, the criteria for evaluating them are similarly indistinct. Indeed, the set of quality criteria that is selected often reflects the authors' view of the nature of the conceptual data model more clearly than any specific definition. Thus, *syntactic correctness* is almost universally included, while others such as, *conciseness* or *flexibility* and *maintainability* are only likely to be regarded as desirable qualities to those for whom the conceptual data model is clearly datalogical.

One of the major strengths of the INTECoM process is the clear delineation of the analysis and design stages and models yet the quality plan, being based on work that had not made this differentiation, did not reflect this. At the end of the analysis phase of the ISPG development, it seemed natural to undertake a quality exercise. While there was nothing to prevent this happening, the plan had not been designed to be used in this way. Thus, for example, a syntactical check of the analysis model had not been envisioned. It became clear that the goals of the two stages should be distinctly recognised, allowing criteria and measurements appropriate to both stages to be determined. The different characteristics and requirements of the analysis and design models had been clearly laid down in the description of INTECoM and a rigorous quality evaluation needed to reflect this.

## ***2. Auditability***

Previous discussion of the NIAM approach to modelling has suggested that the technique can provide a clear audit trail from the initial identification of the fact types to the eventual data structures. Through its incorporation of NIAM and the creation of a set of natural language statements for the design model, INTECoM also has this potential. However no use was being made of this powerful quality mechanism. The ability to trace all constructs in the design model back to their origin would clearly provide a means to identify how, why and when constructs appeared in the models. It would also be possible to identify requirements that had gone 'missing' during the modelling process or had been created without any firm basis in the domain. Such knowledge would be very useful in improving the quality of the final product and therefore a new quality goal, auditability, was recognised.

## ***3. The quality of the process***

The quality criteria, discussed in Chapter 8, were exclusively concerned with evaluating the final product. Consequently, there were no criteria that addressed the process by which the model was constructed. However, this process is highly likely to reflect on the quality of the final product and to this end another quality goal, procedural correctness, was also identified.

Reflection on these issues necessitated a re-appraisal of the quality environment in which the INTECoM development was taking place. Practical experience with the plan suggested two new quality goals and the clear need to formally distinguish the quality measures appropriate to the two models. The straightforward inclusion of the new goals into the existing plan was a possibility but did little to address the remaining issue. Instead, a critical comparison was made between the original quality plan, the inputs defined for the first two stages of INTECoM and the quality framework constructed by Krogstie *et al* (1995). The result of this comparison was the definition of a complete quality framework, illustrated at Figure 23, based on extensions to the Krogstie *et al.* (1995) model. The justification for, and the instantiation of, this framework are discussed in the following section. In addition, several refinements based on quality considerations were also made to the INTECoM framework itself and these are also discussed below.

## The INTECoM Quality Framework

As Krogstie *et al*'s framework suggests, a model is a representation of statements taken from a domain and expressed using some form of formal grammar or language. However, their framework only illustrates a static position and does not recognise the dynamics of model construction. Thus a new concept of *process* is added. This reflects that not only are the constructs of a **language** used to represent the information within a **domain** but that some form of **process** (or method) is also required to build any particular **model**. Lindland *et al* (1994) define the links of **semantics**, relating the model to the domain, and **syntax**, relating the model to the modelling language. Likewise the link of **procedure** relates the model to the process by which it is constructed and gives rise to the procedural quality shown on the new diagram. The goal of procedural quality is that the process by which model construction had occurred is explicit and has been followed appropriately. It is clearly relevant to both phases of model building.

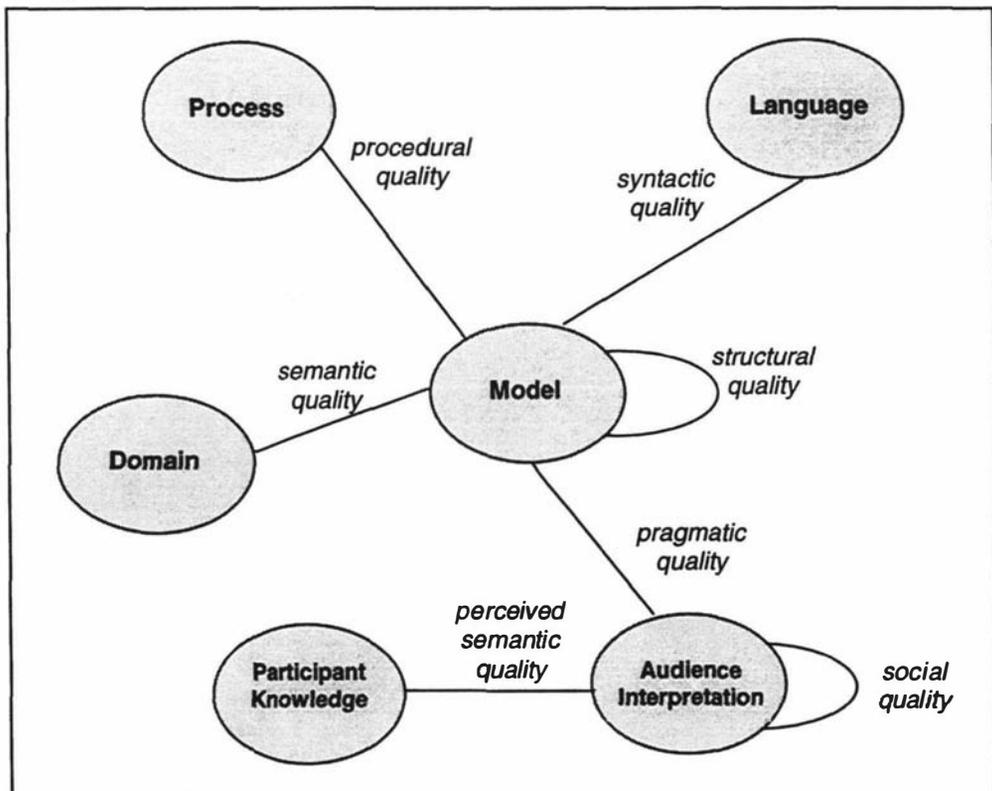


Figure 23 INTECoM - Quality Framework

Kesh (1995), Moody and Shanks (1994, 1998), and Shanks and Darke (1996) identify that the overall quality of a model also requires some desirable structural qualities of the

model itself. This consideration led to the identification of a new quality dimension, that of **structural quality**. The goal of structural quality brings together a number of elements that relate to the inherent soundness of the model and its 'fitness for purpose'. These characteristics are a reflection of a 'good' design and in the context of the INTECoM framework are seen as relevant only to the datalogical design model.

With this new quality framework in place, the first two steps of INTECoM were appraised to identify whether quality goals, compatible with the framework, could be identified. A set of quality goals, focussed on the purpose and inputs of each step, was determined. These are discussed below

## Step 1

### *Purpose and Inputs*

The purpose of Step 1 is to explicitly record the relevant information requirements of the problem domain. Seven inputs to the process are identified: enterprise knowledge, system expectation, analysis procedure, analysis language, syntactic verification, information requirements and semantic verification. Both the analyst and the users are expected to contribute enterprise knowledge to the INTECoM process and this, together with the users' information requirements, suggests that the original quality goals of *perceived semantic completeness* and *perceived semantic correctness* are appropriate. The analyst's contribution of analysis procedure and language suggested the goals of *syntactic* and *procedural correctness*. The combined input of users' and analyst's system expectations suggested a further quality goal of *fulfilled expectations*. The remaining two inputs of semantic verification and syntactic verification did not require quality goals of their own.

### *Qualities*

An initial comparison of these five goals against the qualities identified in the quality framework identified four of the qualities that were not matched: semantic, structural, social and pragmatic. The non-consideration of semantic quality has already been discussed and was not seen as problematic. Structural quality had already been identified as relevant only to the design model and was therefore not a requirement of this step. Social quality was originally considered as pertinent to both steps of the INTECoM method, in which case a goal needed to be identified. However, the decision

to retain discrete individual users' views until the beginning of the design step removed the need to reach a position of agreement between the user views in the analysis stage. Therefore, the lack of this quality in the analysis model was considered acceptable.

Pragmatic quality had not been specifically targeted by the inputs of Step 1 possibly because it is such a fundamental part of the whole step. The choice of NIAM-CSDP for this step necessarily involved the basic principle that all communication with the user was conducted in the users own language, which would thereby be both understandable and understood by them. The decision not to amalgamate the different views also strengthened this aspect by removing from each user the need to understand any view but their own. As such, the need for pragmatic quality was inherent in the entire step, for without it the development could not proceed. However, it is recognised that there could well be advantages in requiring the analyst to specifically determine that the model was both understandable and understood and so these quality goals were added to the set but no additional input was identified.

	<b>Quality</b>	<b>Quality goal</b>	<b>Means</b>	<b>Who?</b>
<b>A1</b>	<i>Perceived Semantic</i>	Semantic completeness	Subjective. User confirmation that their view is complete	User
<b>A2</b>	<i>Perceived Semantic</i>	Semantic correctness	Each fact type has a complete set of correct and relevant examples	User
<b>A3</b>	<i>Pragmatic</i>	Pragmatic correctness a) understandable b) understood	Subjective. Confirmation that infological model is understandable & understood	Both
<b>A4</b>	<i>Procedural</i>	Procedural correctness	Task checklist in place Task checklist complete	Analyst
<b>A5</b>	<i>Syntactic</i>	Syntactic correctness	Fact types are well formed and example set is complete. All objects are defined. All constraints are recorded.	Analyst
<b>A6</b>		Fulfilled expectations	Subjective. Cross check between context diagram and identified fact types. Cross check between identified fact types and initial problem definition.	Both

**Table 12 Quality evaluation of analysis model**

One quality objective, fulfilled expectation, had been identified, however, that did not appear to relate comfortably to any of the qualities of the framework. The initial reaction was to remove this quality goal, as the input seemed to be rather vague and ill defined which would lead to problems in measuring or checking the goal. Nevertheless, removing the input appeared to leave an important gap and it was clear that at least some checking could be undertaken. It was therefore decided to leave the goal in place.

Six quality goals had thus been identified; five of which were closely matched to the quality framework. These goals and their associated qualities are summarised in Table 12 together with the means of checking that a goal had been reached and the agent responsible for this checking. The goals are numbered A1 - A6 for later convenience.

### **Assessment**

The means by which these quality goals could be assessed also required definition. The measures for *perceived semantic completeness* and *perceived correctness* were essentially unchanged from the initial criteria. Users would confirm that their view was complete and that each fact type within their view had a full set of correct and relevant examples. A similar position concerned *pragmatic correctness*, which would also require user confirmation.

No	Task	Quality goal
<b>1</b>	<b>Record system expectation</b>	<b>A6</b>
1.1	Context Diagram	
1.2	Problem Definition	
1.3	User Requests	
1.4	Other	
<b>2</b>	<b>Identify appropriate users</b>	
<b>3</b>	<b>Initial requirement collection</b>	
3.1	Interview	
3.2	Documents	
3.3	Other	
3.4	Initial sentences confirmed	<b>A1, A2</b>
<b>4</b>	<b>Construction of qualified fact types</b>	
<b>5</b>	<b>Confirmation of qualified fact types</b>	<b>A1, A2</b>
<b>6</b>	<b>Collection of examples</b>	
<b>7</b>	<b>Confirmation of example sentences</b>	<b>A1, A2</b>
<b>8</b>	<b>Syntactic verification</b>	<b>A5</b>
<b>9</b>	<b>Expectation matching</b>	<b>A6</b>
<b>10</b>	<b>Confirmation that sentences are understandable and understood</b>	<b>A3</b>
<b>11</b>	<b>Task checklist complete</b>	<b>A4</b>

**Table 13 Analysis Task Checklist**

*Procedural correctness* requires that the process by which the model is constructed is explicitly laid down and that the relevant activities have been completed satisfactorily. Therefore the analyst would be required to draw up a task checklist summarising the essential activities of the step and this together with the completion of the checklist would be taken as the determinant of procedural correctness. The task checklist, with its associated quality goals, developed for the ISPG system is illustrated at Table 13.

One of the tasks placed on the checklist was expectation matching which was defined as a cross check between the infological model and the original high level system requirements as recorded in the system context diagram, problem definition and user requests. Although it is recognised that this only encompasses some of the elements of the system expectation input, it was decided that completion of this task would be taken as *fulfilling expectations*.

All the quality goals derived from the inputs identified by the INTECoM framework and those identified by the development of the quality framework were thus instantiated. The INTECoM framework was, in turn, refined by the addition of two further inputs to Step 1, procedural verification and expectation verification. The amended INTECoM diagram is illustrated at Figure 25 on page 206. Finally, it was noted that all quality evaluation in the analysis step was based on the infological model alone.

## Step 2

### *Purpose and Inputs*

The purpose of Step 2 is to take the users' information requirements, together with any known future requirements and transform them into a potentially implementable data structure. Eight inputs, other than the analysis model itself, had been identified; design procedure and language, enterprise knowledge, previous experience, future requirements, paradigm knowledge, syntactic, structural and semantic verification and these inputs were also considered in relation to the qualities of the framework.

As before, enterprise knowledge, this time with the addition of future requirements, suggested *perceived semantic correctness* and *perceived completeness*, while design procedure, design language and paradigm knowledge again suggested *syntactic* and *procedural correctness*. Paradigm knowledge also suggested that the model should conform to 'good practice', which, in the instance of a relational model, could be interpreted as an appropriate level of normalisation and conformance to various relational design guidelines such as the removal of redundant relationships and non-required relations, and minimal primary keys. These were thus equated with the previously identified goals of the structural quality, *simplicity* and *flexibility*.

Of all the specified inputs, previous experience was the only one that did not appear to be directly related to a specific quality criterion. Although the amount and quality of the

previous experience may well reflect on the overall quality of the designer's solutions, it was considered impractical to attempt to measure this. However, it seemed detrimental to remove the input itself from the INTECoM framework. Nevertheless, it was not included within the quality evaluation. The remaining specified inputs of structural, semantic and syntactic verification had been originally designed to assess some of the identified criteria and, as before, did not require specific quality goals.

### *Qualities*

All these quality goals can be associated with qualities described in the framework; however, three qualities are not represented, semantic, pragmatic and social. Semantic quality was again ignored in favour of perceived semantic quality but the remaining two needed more careful consideration.

The goal of the pragmatic quality is clearly more relevant in this stage where the model not only represents an amalgamation of the user views but also their conversion into relational structures. It had been the intention that the construction of the infological design model through the use of NaLER sentences would provide users with access to the data structures that had been designed. It was clearly important that this was done and that the pragmatic goals of comprehension and comprehensibility were explicitly measured. They are therefore included in the set of quality goals.

The social quality of feasible agreement is also very important in this stage. The designer would have consolidated all the user views into the one global design, incorporated future requirements and restructured some elements in line with relational theory. However, it was not necessary for each user to agree to the complete model. If the designer was able, for each user, to extract NaLER sentences from the design model which corresponded to the original fact types, and if they could be instantiated with the appropriate examples then clearly any apparent conflict between the views was one of perception only. Nevertheless, user confirmation that their view was still supported was essential. Social agreement was thus added to the set of quality goals.

Ten quality goals had thus been identified; all of which were closely matched to the quality framework. These goals and their associated qualities are summarised in Table 14, together with the means of checking that a goal has been reached and the agent responsible for this checking. The goals are numbered D1 - D10 for later convenience.

	Quality	Quality goal	Metrics	Who?
D1	<i>Perceived Semantic</i>	Semantic completeness	Subjective. User confirmation that their view, as shown by NaLER sentences is complete. Confirmation of future requirements	Each user Designer
D2	<i>Perceived Semantic</i>	Semantic correctness	Each NaLER sentence has a complete set of correct and relevant examples. Each 'new' NaLER sentence has a complete set of correct and relevant examples.	Each user Designer
D3	<i>Pragmatic</i>	Understandability	Complete set of instantiated NaLER sentences	Designer
D4	<i>Pragmatic</i>	Understood	Efficient and effective verification User confirmation	Designer Each user
D5	<i>Procedural</i>	Procedural correctness	Task checklist in place and complete	Designer
D6	<i>Procedural</i>	Auditability	All NaLER sentences can be traced to their originating requirement.	Designer Each user
D7	<i>Social</i>	Social agreement	Full view integration Complete set of instantiated NaLER sentences	Designer Each user
D8	<i>Structural</i>	Flexibility	Datalogical model is in BCNF	Designer
D9	<i>Structural</i>	Simplicity	No redundant relationships, primary keys are minimal, all relations are required.	Designer
D10	<i>Syntactic</i>	Syntactic correctness	The datalogical model is relationally correct.	Designer

**Table 14 Quality evaluation of design model**

### **Assessment**

The means by which these quality goals could be assessed again needed to be developed or confirmed. The measure of *perceived semantic correctness* requires user confirmation. In this stage, however, the designer is required to extract the relevant NaLER sentences from the model, instantiate them with examples taken from the analysis model and present the appropriate set to each user. Each user then needs to confirm that the set of sentences and examples is both complete and correct. Once again, *pragmatic correctness* is a required pre-requisite for this activity but can be measured here in slightly more concrete form. A measure of understandability can be assessed by the designer's ability to construct the full set of instantiated NaLER sentences and not only can the user confirm that the sentences have been understood but the designer can also evaluate how efficiently user verification had been obtained.

*Procedural correctness* again required the creation and completion of a task checklist, which is shown at Table 15. An additional measure of procedural correctness was also identified in auditability, which requires the designer to trace the origin of each statement in the model by tracking each NaLER sentence to its source. One additional activity was added to this task checklist. In recognition of 'best practice', peer review at

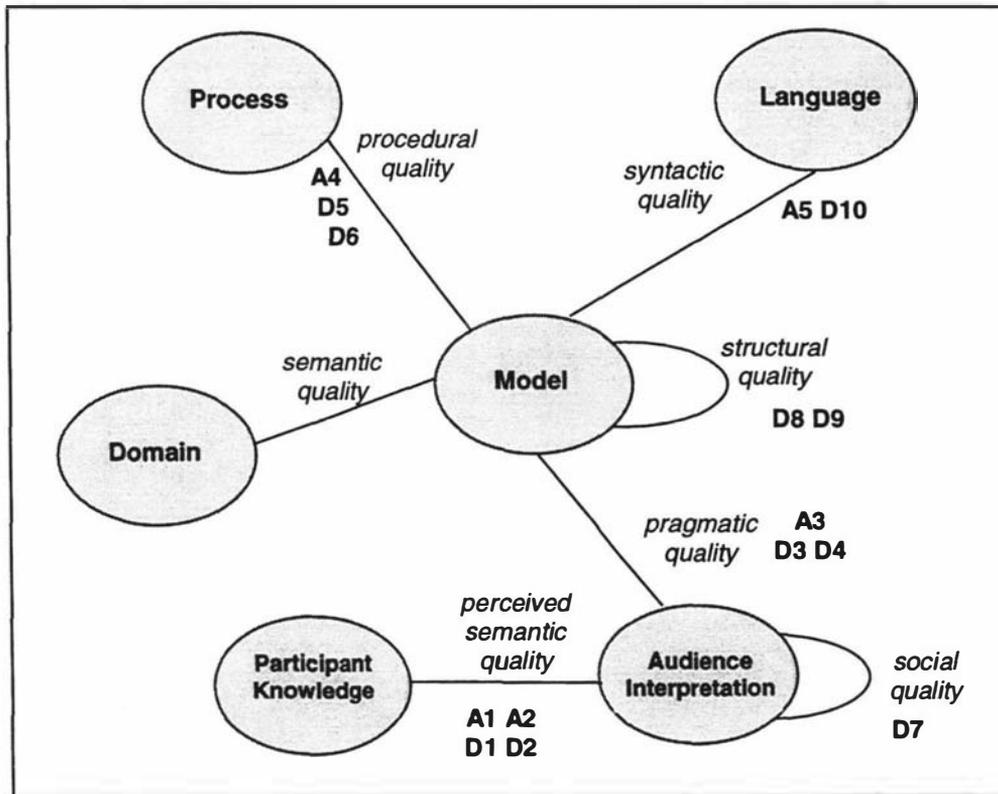
various stages of the evolving design, was considered an essential element of the design process. A formal peer review activity was therefore added to the checklist and incorporated into the activity of Step 2. Social agreement would be determined by the designer's ability to extract the NaLER sentences for each user view and from the users' confirmation that the NaLER sentences and examples represented their original view.

No	Task	Quality Goal
<b>1</b>	<b>Prepare first draft design model</b>	
1.1	Transform analysis models to relational representations	
1.2	Amalgamate logical views	
1.2.1	List all entities and PKs	
1.2.2	Merge entities with same PK	
1.2.3	Check for synonyms	
1.2.4	Check for similar PKs	
<b>2</b>	<b>Generate/evaluate alternatives</b>	
<b>3</b>	<b>Incorporate future requirements</b>	
<b>4</b>	<b>Create final draft design</b>	
<b>5</b>	<b>Verify final design</b>	
5.1	Syntax Check	<b>D10</b>
5.1.1	Create 2-way sentences	
5.1.2	Check participation constraints	
5.1.3	Check PK-FK links	
5.1.4	Check normalisation	<b>D8</b>
5.2	Simplicity check	<b>D9</b>
5.2.1	Check for minimal primary keys	
5.2.2	Check for redundant relationships	
5.2.3	Check for trivial relations	
5.2.4	Check all relations are required	
5.3	Semantic check	<b>D2, D3, D7</b>
5.3.1	Create NaLER sentences	
5.3.2	Populate NaLER with examples taken from analysis model	
5.3.3	Create cross reference table – analysis facts to NaLER sentences	
5.3.4	Check for completeness	
5.3.5	Check for consistency	
<b>6</b>	<b>Audit</b>	<b>D6</b>
6.1	Check source of all NaLER sentences	
<b>7</b>	<b>Peer Review</b>	
<b>8.</b>	<b>User Verification</b>	<b>D1, D4</b>
7.1	Create NaLER user views	
7.2	Correlate NaLER and analysis views	
7.3	Gain user verification	
<b>9</b>	<b>Task Checklist complete</b>	<b>D5</b>

**Table 15 Design Task Checklist**

Whereas the previous goals were based on the infological design model, the remainder was targeted on the (relational) datalogical design. The structural objectives of the model, simplicity and flexibility would be assessed as originally envisioned. Flexibility

would be judged by the level of normalisation, and syntactic correctness would require that the model conformed to relational rules.



**Figure 24 INTECoM - Instantiated Quality Framework**

All the quality goals derived from the inputs identified by the INTECoM framework for Step 2 and those identified by the quality framework were thus instantiated. Figure 24 illustrates the quality framework instantiated with the various quality goals identified by their codes.

### **Quality in Practice - Observations**

A number of the issues pertaining to quality, which arose during the ISPG development, have been addressed in the previous section. In particular, those relating to the differing quality goals of analysis and design models and to the need to focus on the quality of the process and not just the output were seen as important. Addressing these two issues resulted in a fundamental reappraisal of the initial quality criteria and produced a significantly more rigorous approach to quality. However, three additional observations were also made.

### ***1. Self-verification***

The first observation was that one of INTECoM's strengths lay in the fact that the process, which it employed, had a high degree of self-verification. By the end of the analysis phase the users' requirements had been developed systematically from the expression of the high-level information needs and their decomposition into the initial sentences. Everything that appeared in the analysis model had not only been verified by the users but also, to a large extent, explicitly provided by them. Additionally it had not been necessary for the users to have any training in the analysis 'language' as this was exclusively their own natural language, including their own preferred nomenclature. Both the semantic and the articulatory distances, as identified by Hutchins *et al.*, (1985), between the users and the representation of their requirements, was thus considerably reduced.

During the design phase, the creation of the NaLER sentences and examples ensured that both a completeness and a correctness check was undertaken by the developer before the design model was presented to users. Not only did this ensure that a high quality model was presented to users for verification but also ensured that the model was quickly viewed with a high level of credibility. The syntactic check resulting from the input of the NaLER sentences and examples to *InfoModeler*<sup>TM</sup> and the subsequent creation of an optimal normal form relational schema, which matched the design model, also provided an additional quality check. Finally, the user verification at the end of the design phase was also conducted in the users' own language, via the NaLER sentences. Thus, once again, there was no necessity for the users to learn the design language. At no time, were users shown the model in any form other than in natural language sentences. Despite some initial scepticism from two of the users, who were very familiar with E-R/R models, none of the users asked to see the datalogical versions of the models and yet had no hesitation in enthusiastically confirming that their views were understandable, understood, complete and correct.

### ***2. Introduced errors***

The second observation was that there were a number of areas in which errors could be introduced into the models. This was not seen as a problem specific to INTECoM but was a characteristic shared with other database design methods. Further, it was identified that the kinds of errors that could occur were of specific types, e.g. of

omission. No taxonomy of errors had been discovered in the literature and yet this would seem to be an area in which productive work might be undertaken. It was considered possible that if the nature of these errors were better understood and described specific processes or behaviours could be defined that would assist in minimising them<sup>2</sup>. No such taxonomy was attempted but the possibility was noted for future investigation.

### ***3. Quality integration***

Finally it was noted that many aspects of the quality evaluation were part of an on-going process. Quality checks were not generally additional tasks to be undertaken if time and motivation allowed but were integral to the activities within the INTECoM framework itself. Indeed, many of the prescribed activities could not proceed unless previous quality goals had already been satisfied. In other cases, for example the creation of the equivalence tables, a quality check was inherent in the activity being undertaken.

### **Summary**

Undoubtedly, in any information system development, quality matters but as Chapter 8 has indicated no definitive set of quality criteria has been determined for data modelling. Although the construction of the INTECoM framework was not primarily focussed on explicit quality issues, nevertheless it had been largely driven by a desire to identify a quality process for conceptual data modelling, i.e. the intention had been to seek techniques that were 'fit for the purpose' of the activities to which they were applied. In so doing, INTECoM succeeded in incorporating a number of important quality checks into the process that it delineated. Procedural correctness was recognised as an important quality goal and the full potential of NaLER sentences was utilised. Not only was it possible to construct an effective and useable quality framework but also, through such refinements as the task checklists, INTECoM itself was improved.

---

<sup>2</sup> For example the correlation of the NaLER sentences with the analysis 'facts' identified a straightforward error of omission that had occurred in the transformation of the analysis to the design model.

# 13 INTECoM: an instantiation

*“In spite of our best efforts, any formalism we adopt as the basis of the conceptual model, will still be an artificial structure. The concepts will not be perfectly intuitive to anyone; the rules, limitations, and idiosyncrasies will have to be learned. There will be a formal language to be learned, as well as operating procedures.” (Kent, 1978 p.94)*

## Introduction

A theoretical framework for database development, concentrating specifically on the construction of the conceptual models of analysis and design was detailed in Chapter 9. The process of developing a small database, and determining appropriate quality measures, described in Chapters 11 and 12, resulted in various modifications to the original framework. In particular, various strategies, which had not been considered in the initial description, were developed. Although the overall structure of INTECoM has not altered, this chapter consolidates the previous work by delineating the final version of INTECoM, instantiated with elements of the NIAM-CSDP method in the analysis stage and E-R/R and NaLER techniques in design.

The diagram shown at Figure 25 is based on the original diagram on page 133 and incorporates all the modifications. There are three additional inputs in Step 1, procedural verification from the Data Analyst and expectation verification from both the Data Analyst and the Users. Step 2 has an additional input, procedural verification, from the Data Designer while design procedure and language have been amalgamated to maintain the clarity of the diagram. The input of the analysis model to the design process has also changed slightly. On the original diagram it flows from the datalogical view, here it is shown as emanating from the superset. This is to reflect that either or both views may be used and the decision should be at the discretion of the Data Analyst. All other aspects of the diagram are unchanged both in form and content from that shown at Figure 14. Consequently, the detail of many of these elements is not repeated in this description

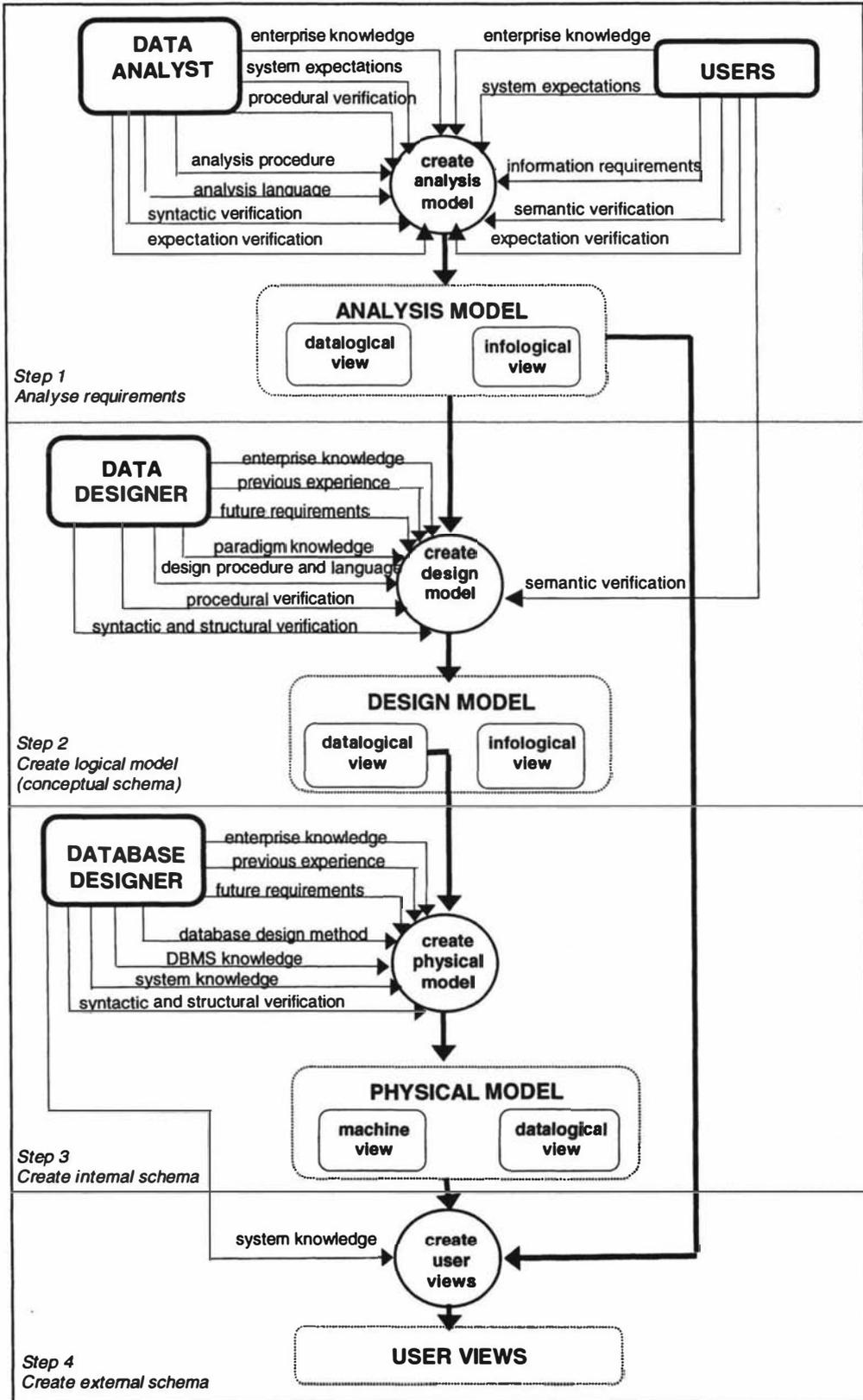


Figure 25 INTECoM Framework – Final Version

## An instantiation of INTECoM

### *Step 1. Analyse requirements*

- **Agents.** There are two identified roles in this step, Data Analyst and User. There may be more than one individual playing each role and, conversely, one individual may play more than one role. While the identification of suitable individuals for the first role is likely to have been completed prior to the start of development, the identification of appropriate users is seen as a responsibility of the Data Analyst and, thus, appears as a task within the analysis process. It is necessary for the Data Analyst to be familiar with the language and procedure of the chosen analysis technique and it is desirable, but not essential for the Data Analyst to be familiar with the problem domain. However, it is not necessary for the Data Analyst to be experienced in database implementation techniques. Likewise, it is not necessary for the User to have any specific technical training or understanding. In this chapter, a capitalised rendition of the agent names is used to signify that it is the role that is being referred to rather than any specific individual.
- **Inputs.** There are twelve identified inputs to the analysis process. Five are concerned with verification, of the procedure, syntax, semantics and expectations. Four relate to pre-existing knowledge that the agents bring to the development activity, i.e. enterprise knowledge and the skills required by the Data Analyst. Two are concerned solely with the expectations that both the Data Analyst and User role bring to the development. The final input, information requirements, contains the domain knowledge that the Data Analyst needs to elicit from the User. This is the most significant input into the following process.
- **Process: Create Analysis Model.** This process is based on the steps of the NIAM-CSDP, summarised on page 73. However, the steps have been adapted to make the most productive use of the *InfoModeler*<sup>TM</sup> CASE tool. While the following process is described in a linear fashion, it is recognised that many of the tasks and activities may be performed iteratively and/or concurrently, at the discretion of the Data Analyst. The activities required by this process are summarised in Figure 26.

1. Construct task checklist
2. Record expectations
3. Identify appropriate users
4. Collect initial information requirements
5. Construct qualified fact types
6. Collect example sentences
7. Verify syntax
8. Confirm fact types are understandable and understood
9. Confirm fact types are correct and complete
10. Confirm expectations are being met
11. Confirm task checklist is complete

**Figure 26 Create Analysis Model - Activities**

- 1. Construct task checklist.** The Data Analyst will create a task checklist, which defines the procedure to be followed in this process. The detail of the list will depend on the analysis technique and CASE tool to be used. It may well remain the same from one development to another. The tasks described here are based on the checklist developed for the ISPG development detailed in Chapter 11.
- 2. Record expectations.** The recording of expectations can take a number of forms and many will come from other areas of the system's development. These may include a set of user requests, a feasibility study, a context diagram, and problem definition statements. Although there is nothing to preclude it, there is no requirement for the Data Analyst to create these documents directly, merely to collect together the information that they contain. This record forms the basis for checking that expectations have been fulfilled (quality goal A6).
- 3. Identify appropriate users.** The Data Analyst is responsible for identifying a set of appropriate users. There should be at least one user from each user community with an interest in the final system. It should generally be sufficient to document a single view for each user community rather than for each individual. However, there is no reason, apart perhaps from duplication of effort, why individual views cannot be constructed. Indeed, this may be desirable where it proves difficult to capture a consensus view from one

community. The decision on the number of views that are required is the responsibility of the Data Analyst.

4. **Collect initial information requirements.** This is likely to include a number of different activities chosen from the usual range of requirement elicitation techniques which may include, interviews with users, and a study of input and output documents. For each user view that is to be constructed, a set of initial sentences is collected, preferably with the assistance of the user representatives.
5. **Construct qualified fact types.** This activity transforms the initial sentences into qualified fact types suitable for input to the CASE tool, following the guidelines of NIAM-CSDP.
6. **Collect example sentences for each qualified fact type.** This activity provides a full set of examples for each of the fact types identified in the step above. It is probable that activities 4, 5 and 6 will occur concurrently. As an initial sentence is formulated, a qualified fact type can often be postulated and confirmed by the creation of examples. A CASE tool such as *InfoModeler*<sup>TM</sup> encourages the Data Analyst to input the fact types and examples together.
7. **Verify Syntax.** The Data Analyst is required to confirm the soundness of the fact types and to reflect on alternatives. Once, the Data Analyst believes that a complete set of fact types and examples has been collected for a particular user, the model for that user should be checked for correct syntax. This will be repeated for each user view. This can usually be achieved automatically if a CASE tool is being used (quality goal A5).
8. **Confirm that sentences are understandable and understood.** Each user needs to confirm that the sentences from their view are understandable and understood. Some sentences may need explanation but generally the discussion of examples is sufficient to clarify understanding. It is essential that the Data Analyst ensures that this understanding had been achieved, for without it the confirmation required by activity 9 is not valid (quality goal A3).

- 9. Confirm qualified fact types and examples.** The Data Analyst, from the syntactically checked model should compile a complete set of fact types, instantiated with examples. Again, the use of a CASE tool such as *InfoModeler*<sup>TM</sup> significantly simplifies this process. This list of natural language sentences, possibly with additional information on the composition and description of each object, is then provided to the user whose view it represents. The user is asked to confirm that the sentences are correct, complete and that the examples are representative and valid (quality goals, A1 and A2).
- 10 Confirm expectations are met.** This activity falls into two parts, one undertaken by the Data Analyst and one by the Data Analyst and User together. Initially, it is the responsibility of the Data Analyst to compare each user view to the initial requirement documents and check that the two are compatible. Any single view will not necessarily support all the requirements identified for the system but all requirements should be accounted for by inclusion in at least one view. Any omissions should be included, if necessary by the development of an additional view, that of the Data Analyst. This may well be necessary where the system incorporates a number of new or changing, requirements that are not readily identified by the User. It may also be useful to identify any fact types that have been included in a user view but do not appear to have been required. The User, in conjunction with the Data Analyst, also needs to confirm that the original requirements, as laid out in the original documentation are being met (quality goal A6).
- 11. Confirm task checklist is complete.** The final activity in Step 1 is for the Data Analyst to verify that the task checklist is complete and that the process has been followed as originally planned. Any modifications made to the plan should be noted and justified (quality goal A4).

The analysis process is then complete and the following outputs have been created.

- **Infological Analysis Model.** This model is the superset of the formal, natural language deliverables created for the user views. For each view, it will consist

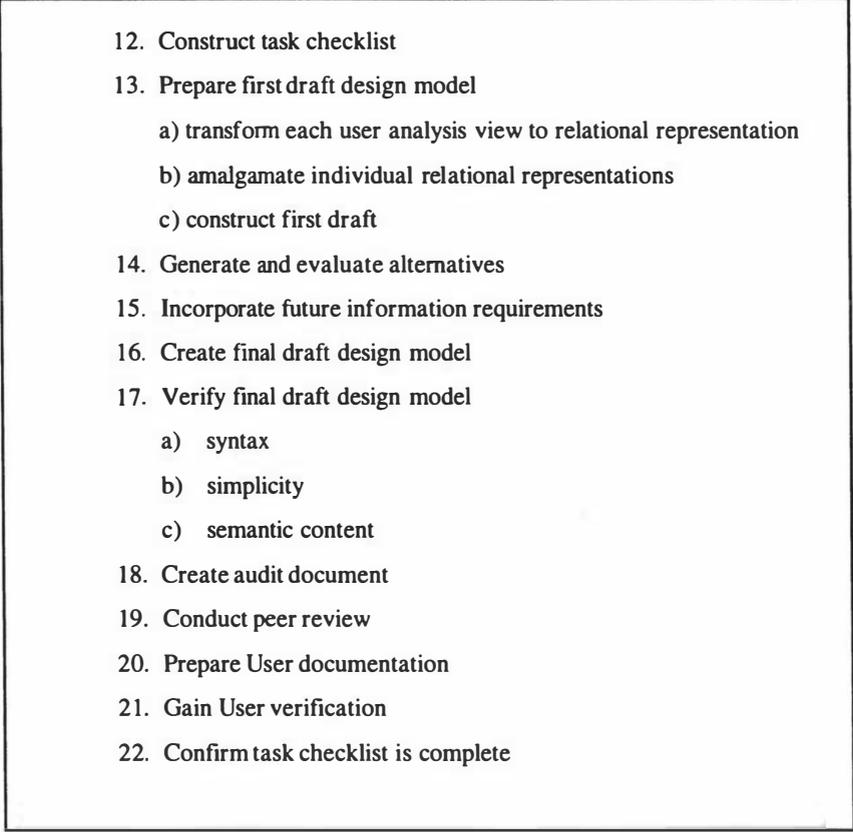
of the complete set of syntactical correct fact types, example sentences, object descriptions and constraint descriptions.

- **Datological Analysis Model.** This is the superset of annotated ORM diagrams. When *InfoModeler*<sup>TM</sup> is used, these diagrams, one for each user view, are created automatically as the natural language information is being input. However, there are some constraints that can only be entered by annotating the diagram. In the worked example described in Chapter 11, the datological analysis model was not used except to enter such constraints.

### *Step 2 Design the logical model (conceptual schema)*

- **Agents.** There are two roles identified for this step Data Designer and User. However, it is expected that there would be considerable liaison between the Data Analyst and Data Designer and that the individuals acting in the user role in the previous step, would continue to do so. The Data Designer will require a good understanding of the database paradigm, e.g. relational theory, and expertise in the design language and procedures. However, as in the previous step, there is no requirement for the User to have any specific technical knowledge.
- **Inputs.** There are nine inputs identified for this step, all but two of them coming from the Data Designer. Three inputs relate specifically to quality checking, i.e. semantic, syntactic and procedural verification. One input, future requirements, relates to the information requirement and can extend the scope of the development. Four inputs relate to the specific skills of the Data Designer, paradigm knowledge, design procedure and language skills, and previous experience. It is the application of these latter inputs to the final one, the analysis model itself, that comprises most of the activity of the following process.
- **Process: Create Design Model.** The process described here is based on various aspects of traditional E-R modelling together with the NaLER technique. However the primary entities and relationships are directly identified from the analysis model rather than from any independent analysis activities. While the following process is described in a linear fashion, it is recognised that many of the tasks and activities

may be performed iteratively and/or concurrently at the discretion of the Data Designer. The activities required in this step are summarised in Figure 27.

- 
12. Construct task checklist
  13. Prepare first draft design model
    - a) transform each user analysis view to relational representation
    - b) amalgamate individual relational representations
    - c) construct first draft
  14. Generate and evaluate alternatives
  15. Incorporate future information requirements
  16. Create final draft design model
  17. Verify final draft design model
    - a) syntax
    - b) simplicity
    - c) semantic content
  18. Create audit document
  19. Conduct peer review
  20. Prepare User documentation
  21. Gain User verification
  22. Confirm task checklist is complete

**Figure 27 Construct Design Model - Activities**

1. **Construct task checklist.** The Data Designer will need to create a task checklist, which defines the procedure that will be followed in this process. The list will be dependant on the design technique that will be used and the preferences of the Data Designer but may well remain the same from one development to another. Again the tasks described here are based on the development undertaken as part of this research.
2. **Prepare first draft design model.** This activity comprises two major stages. Either, or both, the infological or datalogical view of the analysis model may be used as the starting point at the discretion of the Data Designer and may well depend on the technical support that is provided by the analysis CASE tool.

- a) **Transform each analysis view to a relational representation.** Where the analysis model had been recorded in a CASE tool this transformation can be effected quickly and automatically by the tool itself. If a suitable tool is not available, the appropriate transformation algorithm is detailed in Halpin (1995) and can be applied manually. Either of these methods will result in optimal normal form relations. The transformation is effected independently for each user view.
- b) **Amalgamate user views.** The relational representations are used as the basis for this amalgamation. It is important that all views are considered and all aspects included initially. The following strategy proved effective in the worked example.
- i) Create a table, in a word processing document, which for all views lists all the entities, their primary keys and the source view.
  - ii) Sort this table on the basis of the primary keys.
  - iii) Identify and merge entities with identical primary keys. The order of the primary key columns is not be significant.
  - iv) Identify synonyms and homonyms and merge entities, if appropriate. The example sentences can be used to identify merger candidates.
  - v) Identify and merge, if appropriate, entities with similar primary keys. These are the most difficult merger candidates to identify. If there is any doubt, and the examples are unclear, it may well be prudent to retain their individual identity until later in the design process.
- c) **Construct first draft design model diagram.** Apart from the initial relational diagrams, all the work above will probably have been undertaken without the use of diagrams. The construction of a graphical representation at this point, while not essential, can be of assistance in visualising the overall shape of the emerging data structure.

This completes the preparation of the first draft design model.

3. **Generate and evaluate alternatives.** This is the most creative aspect of the process and not an activity that can easily be prescribed. The Data Designer will evaluate

the first draft design model and seek to make ‘improvements’ based on previous experience, paradigm knowledge and an understanding of the development context. Any number of alternatives may be generated for all or part of the model and peer review is likely to be one useful means of arriving at satisfactory decisions. Merger candidates identified, but not resolved, in the earlier stage can be usefully revisited here.

4. **Incorporate future requirements.** The incorporation of known future requirements may already have been addressed in the previous activity. If it has not, then the preferred model derived in that activity will be need to be checked and adapted, if necessary, to accommodate all such requirements. Even where no future requirements are known, the Data Designer has a responsibility to ensure that the design retains reasonable flexibility. This is not an aspect of a model that can easily be measured, except perhaps with hindsight, nor is it an activity that can be prescribed. The Data Designer’s experience and skill will play a significant part in determining the level of flexibility that can be achieved.
5. **Create final draft design model.** The work to date now needs to be formally consolidated into the final draft design model. This should comprise the usual E-R deliverables of a diagram and a full supporting data dictionary. It will provide the basis for the final verification tasks.

This completes the preparation of the datalogical view of the final draft design model.

6. **Verify final draft design model.** This comprises three major activities, each of which consists of a number of tasks.
  - a) **Check model’s syntax.** A systematic and formal check on the syntax of the final draft model ensures that no syntactic errors or anomalies are present when the model is finally submitted for user verification (quality goal D10). This check confirms that the data structure is being constructed ‘correctly’. There are three tasks required by this activity.
    - i) Check the model for conformance to normalisation rules. The level of normalisation is at the discretion of the Data Designer but a minimum of Boyce Codd Normal Form is recommended (quality goal D8).

- ii) Check that the primary to foreign key links are correct, appropriate and represented accurately on the diagram. The normalisation check should identify any such problems but a final check is recommended.
  - iii) Check participation constraints. The construction of 2 way sentences may assist in this process. However, the construction of NaLER sentences and their instantiation with examples will also help to confirm that optionality has been correctly established.
- b) **Check model for simplicity.** This is designed to ensure that the model conforms to 'best' practice. It is recognised that the final decision regarding checks ii) to iv) will be made by the Database Designer in Step 3. Some apparently redundant or trivial elements may, therefore, remain in the model until that time (quality goal D9). There are four tasks required by this activity.
- i) Check for minimal primary keys. In conformance with relational theory, each primary key should contain the minimum number of attributes to ensure unique identification of each tuple.
  - ii) Check for redundant relationships. It may be correct to carry more than one relationship between two relations, although such instances should always be checked. There may also be derivable or spurious relationships that can be eliminated.
  - iii) Check for trivial relations. Some 'key only' or minor 'look-up' relations may be considered trivial and removed from the model. However, this is generally a decision best left to the Database Designer in Step 3.
  - iv) Check all relations are required. Again, the Database Designer may be in a better position to make such decisions. However, any relations that are clearly redundant could be removed at this point.
- c) **Check model for accurate semantics.** This activity requires the Data Designer to check that the semantics represented in the model are correct and complete in terms of the initial statements of information requirements. The creation of the NaLER sentences and their cross-reference to the facts of the analysis model is a powerful tool for the Data Designer to confirm that the 'right' database is being

constructed (quality goals D2, D3, and D7). There are four tasks required in this activity.

- i) Create a full set of NaLER sentences. Each element on the model, apart from the relations, will have an equivalent NaLER sentence.
- ii) Populate NaLER with examples. Every NaLER sentence will have a set of example sentences constructed, as far as possible, from the examples provided during analysis. Where no examples exist, the Data Designer will need to construct suitable examples, perhaps from existing documents or from additional User input. Wherever possible, new examples should be confirmed as valid by a member of the user community.
- iii) Create cross-referencing 'equivalence' tables one for each original user view. These tables show how each analysis fact type is represented by one or more NaLER sentence. For those sentences which cannot be matched to one or more analysis facts, the source of the sentence should be recorded, e.g. as a result of normalisation, determined by a future information need etc. An important principle underlying this task is that all NaLER sentences can be traced to their point of origin and that all analysis facts are represented, in some form, in the design model.
- iv) Check for completeness and consistency. Construction of the equivalence tables should identify all missing or misrepresented facts. However, a final formal check is recommended.

**7. Create audit document.** The previous verification tasks will have created the audit trail required in this activity. The source of all NaLER sentences will have been noted. No element will thus be present in the design model that cannot be traced to its source (quality goal D6). If the previous tasks have been fully completed, this activity only requires that an audit document be compiled from the information available.

**8. Conduct peer review.** A number of peer reviews may already have taken place. However, a formal peer review of the complete model, undertaken prior to user verification, is recommended. Ideally, this review should consider the infological view and the audit trail as well as the data structures.

**9. Create User documentation.** While a full set of sentences and examples will have been created in previous activities, it is useful to create a subset for each user view. This ensures that each user is only required to verify their own equivalent set. Each set of user documentation should include:

- the original set of analysis fact types and examples for that user's view,
- a cross-reference table of equivalencies, showing the mapping between each fact type and its NaLER equivalent,
- a set of NaLER sentences and examples for that user's view, and
- a full set of all NaLER sentences for reference.

One copy of this documentation should be given to the User and another retained for use by the Database Designer as a basis for the creation of physical User views in Step 4.

**10. Gain User confirmation.** This activity requires each user, for whom an analysis view was constructed, to verify that the NaLER sentences correctly represent their view (quality goals D1 D4). This can be assisted by encouraging the user to compare the initial analysis sentences to the NaLER sentences. Many will be a straightforward match and can be confirmed easily. For the remainder, the cross-referencing equivalence table and the use of examples can help clarify how an initial information need is being represented in the design model.

**11 Confirm task checklist is complete.** The final activity in Step 2 is for the Data Designer to verify that the task checklist is complete and that the process has been followed as originally planned. Any modifications made to the plan should be noted and justified (quality goal D5).

The design process is now complete and the following outputs have been created.

- **Infological Design Model.** This model is the superset of the natural language statements and associated deliverables created for the user views. For each view, it will consist of the relevant set of NaLER sentences, example sentences and equivalence tables.
- **Datalogical Design Model.** This model consists of a standard E-R/R deliverable, i.e. a labelled diagram and full supporting data dictionary, together

with associated deliverables. These may include a record of design decisions taken and estimates of entity volumes. The number and extent of these deliverables will be at the discretion of the Data Designer in consultation with the Database Designer.

### *Steps 3 and 4*

Consideration of these steps is beyond the scope of this study and, therefore, further discussion is not appropriate here.

### **Summary**

This chapter has provided an example of an instantiation of the revised INTECoM framework. However, it is important to note that the framework is not designed to function only around these two techniques. Indeed, it should be possible to slot other modelling facilities into the appropriate steps. Clearly, some of the processes, strategies and quality measures would need to be adapted to accommodate alternative methods. Nevertheless, the fundamental framework and quality goals should remain intact.

Whatever techniques or methods are used to instantiate INTECoM, there are three principles, on which the framework is founded, which should not be abandoned. Firstly, there is the axiom, taken from NIAM, that all communication with the User should be in that user's own language, i.e. in some form of natural language. The use of NIAM-CSDP and NaLER, demonstrated in this example, ensures that this is the case. Secondly, the clear distinctions between the analysis and design stages, and between the infological and datalogical views of the models, should be preserved. In the first instance, it is important, regardless of the names of the stages, that the differing behaviours of analysts and designers should be recognised and encouraged at appropriate points in the development cycle. In the second instance, the particular purposes of the different views should be acknowledged and met separately. It should also be ensured that the two views are tightly coupled and are effective representations of the same underlying model. The third principle is that of auditability. It should be a common expectation that any constructs in a design model can be tracked to their source and that a designer is accountable for ensuring the semantic integrity of a model.

This chapter, and the practical work on which it is based, has shown that with the introduction of NaLER, it is possible to adhere to these principles using current techniques. Other techniques, perhaps better suited to INTECoM, may exist, and the further testing of the framework remains a challenge for the future. Therefore, the outline provided here should be taken as one successful instantiation of the INTECoM framework rather than a definitive description of a conceptual data modelling method.



# 14 Implications

*“If words were nuts and bolts, people could make any bolt fit into any nut: they’d just squish the one into the other, as in some surrealist painting where everything goes soft. Language, in human hands, becomes almost like a fluid, despite the coarse grain of its components.” (Hofstadter, 1979)*

## Introduction

INTECoM is a comprehensive framework, in that it covers all recognised stages of the database design activity. However, the focus in this study has been on the conceptual modelling activities of the first two stages, i.e. the analysis of the user requirements and the logical design of a suitable data structure to support them. While this is, arguably, a rather wide definition of ‘conceptual data modelling’, it is not inconsistent with the broad range of definitions used by both researchers and practitioners in the area. It also corresponds to the ‘conceptual’ and ‘paradigm’ models of the meta-data architecture discussed on page 35. Although there may be superficial similarities between INTECoM and other database design methodologies, there are a number of significant implications, for both practitioners and educators, inherent in its adoption.

## Implications for Practice

In practice, perhaps the most significant effects of the use of INTECoM would result from the clear demarcation between the stages of analysis and design. It is generally accepted in other areas of information systems development that the different stages will require different activities and techniques (Avison & Fitzgerald, 1995). However, this differentiation is not apparent within data modelling where the distinction is seen largely in the degree of detail, rather than as an inherent difference. The tacit acceptance of a situation that provides a single technique to fulfil the functions of both stages is both widespread and potentially dangerous.

### *Analysis and Design Demarcation*

INTECoM recognises the differences in analysis and design activities and seeks to match those activities with appropriate techniques. In so doing, it also seeks to minimise the amount of design decisions made early in the process by utilising a technique which defers such decisions until the end of the analysis stage. As has been argued previously, the use of E-R Modelling for recording the results of analysis requires some fundamental design decisions to be made at the outset: decisions that may never be revisited. During the analysis stage of INTECoM the focus is very clearly on understanding and recording the data requirements of each user, in a form that is accessible to those users without the need for specific IS skills. As a result, the behaviour of the data analysts, and the background and skills that they require, are clearly differentiated from those of the data designers.

### *Clarification of the Analyst Role*

Data analysts using the INTECoM framework could be expected to interact with appropriate users or user representatives, on an individual basis, and be specifically interested in uncovering and recording a complete and accurate view of the users' perceived needs. Using the NIAM-CSDP as an analysis tool, there is no requirement to reconcile those views, as they are collected, but only to record each one accurately. The analyst needs to be skilful in assisting the user to identify relevant needs and to provide appropriate examples. Interpersonal skills are likely to be highly valued and a good understanding of the organisation's business is likely to be more useful than a detailed understanding of database theory.

### *Analysis Consistency*

As the NIAM-CSDP provides a significantly more prescriptive approach to analysis than E-R Modelling it should be possible to ensure fairly consistent results from different analysts. This is an important consideration in co-operative situations, reducing the amount of re-work required in integrating a number of individual models in large applications and reducing the amount of inappropriate creativity which may be introduced by individual categorisation at too early a stage. It would, therefore, be reasonable for the analysis to be conducted by a number of different people, even in

geographically remote locations, as the resulting fact types, which need to be well supported by appropriate examples, will be consolidated and integrated at a later stage.

### *Clarification of the Designer Role*

On the other hand, the second stage of INTECoM would begin with a set of clearly documented, individual user requirements. Although additional requirements may need to be addressed by the designer, in most situations, a large part of the analysis work will have been completed. The designer is thus able to focus on creating appropriate data structures to support specific requirements. As has already been suggested, the skills required by a data designer include a thorough understanding of the database paradigm for which they are designing, together with a flair for innovative problem solving. The primary focus of the designer is thus not on interacting with the users but on transforming their documented requirements into appropriate database structures. While designers undoubtedly have a responsibility to the users, it is primarily to their technical colleagues that they should be required to justify their designs and decisions.

### *User Accessibility*

Another benefit, implicit in the adoption of INTECoM is an increase in user accessibility, both to the process and to the models themselves. Simsion and Shanks (1993) observe that the more experienced modellers in their study were far more likely to use constructs, and names for those constructs, which appeared to have little grounding in the scenario description they had been given. It was clear that such modellers were using past experience and re-using previous patterns to a considerable extent, rather than focussing on the specific information requirements of the users. This situation, with its inherent dangers (Yunker, 1993), would seem to be characteristic of expert designer behaviour but runs the risk of only capturing the modeller's view of the requirements. The designers' brief, after all, is to create an optimum solution. However, this behaviour requires a considerable degree of faith on the part of the users, to whom it may not be clear whether or not the unfamiliar names and constructs will support their information needs.

The use of the infological models in both stages of the INTECoM framework is intended to alleviate this alienation and increase the users' level of confidence in the personnel, the process, and the models themselves. Anecdotal evidence suggests that it

is possible “when analysts explain a graphical model to users” for everyone to “agree that an incorrect model is correct” (Sharp, 1994 p.D3). This may arise, as Marche (1993) suggests, because “people are so effective at intuitively” accommodating ambiguity or confusion in a model by “projecting meaning onto the data structures rather than abstracting meaning out of them” (p.45). The use of a formalised subset of natural language, as, from the users’ perspective, the language of analysis and design, improves the chances of accurate validation and releases users from the need to understand technical jargon and techniques. It thus increases the potential for positive user involvement. The conceptual models created during the analysis and design stages provide one of the most important foundations for the development of an information system; errors in them may not be uncovered until much further through the development life-cycle and may be costly to correct. For this reason, if no other, it is essential that user access to them should be as straightforward as possible.

Even without the full adoption of the INTECoM framework, the use of a technique such as NaLER, designed to provide a full, natural language interpretation of a design model, can be of considerable benefit. NaLER, after all, is primarily a more structured way of recording the informal explanations that frequently take place between designers and users. The ability to compare design statements with the original sentences, and particularly to see them instantiated with the same examples, adds considerable value to the activity. However, even just the provision of an intelligible, complete and accurate interpretation of a design model is likely to be welcomed by non-technical users.

### ***Flexibility***

The existence of a distinct analysis stage also provides a much greater degree of flexibility than a process that views analysis and design as a hybrid activity. The INTECoM infological analysis model should be much less implementation-oriented than the more traditional analysis data model. If there were to be a significant paradigm shift in database technology, equivalent to the advent of the Relational Model in the late 70’s, many currently constructed conceptual data models would become obsolete. By avoiding the use of quasi-relational constructs such as entities, the infological model contains far less implementation bias than its E-R/R counterpart. Such a model would

thus be better positioned to become the basis for non-relational database designs and to take advantage of the meta-model independence described in Chapter 3.

There are other reasons too, for moving away from the relational dependence exhibited in many conceptual data models. As Kent (1978) suggested it might, a significant investment has been made in educating users to think of their data in relational terms and in persuading them to believe that that is how their data needs to be structured. A change in paradigm would result in a lot of confused, if not disaffected, users. Additionally, by excluding relational bias, it becomes easier to design mixed paradigm databases from the same conceptual model.

### ***Fast Track Development***

On the other hand, there are situations where a straightforward relational implementation is all that is required. The use of an analysis technique appropriate to the first step of INTECoM, such as the NIAM-CSDP, could be beneficial where there are uncontentious database applications awaiting development that would benefit from rigorous but essentially mundane analysis but which require little, if any, innovative design. In these situations a relatively inexperienced modeller, with the assistance of a CASE tool such as *InfoModeler*<sup>TM</sup> may well be able to produce a competent database solution. The expensive, creative and innovative problem solving of the design stage may not always be appropriate or necessary.

### ***Quality Control***

The use of the INTECoM framework also provides a practical opportunity for quality control, not only of the models themselves but also of the process whereby they are constructed. Amer (1993) observes that conceptual modelling errors can signify errors in database processing. This is particularly likely to occur if the modellers are unable to account for the use of constructs in their model. The 'way of controlling', as Bronts *et al.* (1995) describe it, has not been a central focus of this study, however, it has been suggested that INTECoM provides more in-built checks than the traditional database design process. The existence of individual user requirements, clearly documented in accessible language, together with the expectation that the designer will provide an infological view of the design, combine to offer opportunities to track the means by which specific requirements are being met or conflicts resolved. When INTECoM is

operated as suggested, a two-way audit trail is constructed for every statement of user requirement and every element of the final design, providing a means of accountability that is not possible using traditional methods. As the worked example illustrated, a rigorous quality framework can be constructed and operated for an INTECoM development with very few tasks that are additional to normal 'best practice'. Quality checking, rather than imposing an overhead on the activity, becomes an integral part of it.

### *Increased Timescale*

A potential criticism of INTECoM could be that the division of the conceptual modelling activity into two discrete stages would increase the overall time required by the database design activity. However, the use of a prescriptive technique during the analysis stage may well make it easier to provide a realistic time estimate, as well as shortening the overall time by allowing a number of analysts to work simultaneously. A significant time can be spent, in a traditional analysis phase, in attempting to resolve 'stakeholder' conflicts and in building the definitive model of user requirements. The strategy, of recording only individual requirements and not integrating them until design, could well reduce the overall time requirements of the analysis phase.

Clearly, some of the time saved will be required to integrate the views in design but the provisional datalogical design model which is produced early in the stage will be soundly based, thus facilitating other aspects of information systems development. While it is possible that some activities required by the use of INTECoM may take longer, it is envisaged that others, particularly those that relate to user verification, could take considerably less. It is also likely that rework, necessitated by inaccurate specification, or misunderstanding of user requirements, will be minimised.

### *CASE tool support*

Another criticism could be that there is no holistic CASE tool support for this instantiation of INTECoM, and it is accepted that elements of the NIAM-CSDP, such as the transformation algorithm, can be difficult and tedious to undertake without automated assistance. While it is possible to overcome these issues by using two CASE tools in combination, as Grimes (1998) suggests, this interrupts the smooth transition from analysis to design and introduces unnecessary duplication and complexity. The

provision of a single integrated CASE tool would greatly increase the attractiveness of the framework.

### ***'Expert' Resistance***

Experienced data modellers who have developed a number of strategies for successfully completing complex and wide-ranging hybrid activities may not welcome the division of roles and responsibilities that are implicit in the INTECoM approach. Experienced modellers' skills have been developed in response to the need to resolve many of the conflicting issues discussed in this study, and will have been learnt experientially over a number of years. 'Experts' have no reason to embrace a simplification of the process that they have made their own. Indeed, they are likely to have a vested interest in not doing so! However, the adoption of INTECoM is not primarily targeted at those individuals or organisations that are confident in their ability to successfully capture and structure their data requirements. INTECoM is intended to provide a simpler, but effective, process for those who currently do not have this capability.

### **Implications for Education**

The adoption of INTECoM would also have implications for educators, both of students and professionals. Hitchman (1995 p.31) refers to "unsupported assertions that educators find the entity-relationship approach difficult to teach", while Pletch (1989) observed that "students generally find it difficult to manage the complexity surrounding the conceptual model, its development and usage" (p.74). With the INTECoM approach, the aim, initially, would be to provide students with the ability to complete the tasks of the analysis stage and the creation of the first draft datalogical design model.

### ***Prescriptive Method of Analysis***

A prescriptive approach should be easier both to learn and to teach than a descriptive one, particularly if it is supported by an appropriate CASE tool<sup>1</sup>. An holistic understanding of the entire process is not required before progress can be made, although clearly the more complete the understanding the more easily and intelligently

---

<sup>1</sup> Everest (1994) considers that the availability of InfoModeler is an important consideration in the ability to teach NIAM-CSDP successfully.

the process can be followed. The users of such an approach can begin to follow certain steps with minimal instruction. They can also be taught to identify mistakes in executing the steps, even before they are able to grasp the full implications of those errors, thereby satisfying Eden's (1996) criterion that "novices need a well defined process with validity tests at each stage" (p.42). The ability to transform the results of analysis into an implementable solution also provides timely feedback and an opportunity to study the effect of 'design' on such results. Consequently, novices have much less need for trial and error. Instead, particularly where the UoD is not overly complex, novices can be expected to create reasonably accurate solutions fairly quickly: an important consideration in helping to build their self-confidence.

In contrast, the current teaching of modelling skills is often a process of providing examples of input and output and then encouraging students to induce their own rules for constructing a conceptual model. Given the difficulty of recognising the 'best' solution for anything other than trivial examples, together with the obvious problem of determining what criteria a student has used for classification and consequently how a student has arrived at a solution, it is difficult for an educator to provide any form of objective assessment or feedback. In addition, students are required to take a holistic view from the outset, learning to identify potential relations during their analysis and to represent the users' view in the relational paradigm, while they are still coming to terms with the paradigm themselves. As Sharp (1994), comments the students are "learning how to create graphical models that are targeted at being understandable by other analysts and potentially not understandable by users" (p.D3) and, it could be added, not always by the students themselves.

Sharp (1994) reporting on personal experiences of teaching NIAM to practitioners, comments that, before such instruction, "staff with modelling skills are not confident of those skills" and reports that if such students are asked to create a model of a relatively simple problem using their preferred method, only "...one fourth or less have a credible start of a model after 20 minutes" (p.D3). Petch (1989) also observes that modelling students often display the same characteristics of high school math students who "sit and look at the words in the text for some time waiting for the required equation to be miraculously revealed to them" (p.74). After instruction in NIAM-CSDP, Sharp (1994) observed an increase in confidence among students and suggested it came from now

having a “specified approach for attacking each problem” and noticed that “the mental activity that [*had previously*] delayed a solution to the problem could usually be overcome” (p.D11).

Providing a prescriptive approach to the initial stage of conceptual data modelling may well lead to a decline in the creativity of the process but in common with most crafts, expertise comes with prescriptive experience. Most, if not all, creative professions require novices to spend a considerable amount of time learning, prescriptively, the fundamentals of their chosen discipline. As an appreciation begins to develop, of the subtle interplay between process, content, environment and result, a craft apprentice may begin to experiment with alternatives, moving towards creativity, innovation and originality from a solid foundation. Using INTECoM, a similar development is available for novice modellers. When learning any new technique, the need for creativity and the provision of a multitude of possibilities, can be confusing and counter-productive, both hindering the learning process and lengthening the learning curve. Initial experience with a prescriptive approach can increase self-confidence, create an appreciation of the processes involved, improve productivity and allow for gradual and subtle shifts towards creativity in the final product in a way that does not compromise its fundamental soundness.

### ***Reduced Emphasis on Teaching Design***

With the instantiation of INTECoM described in this study, novices would be expected to learn the basics of the E-R/R approach for use in the design stage. However the expectation would be that the refinement of those skills would come through experience, exposure to a range of real situations and preferably the opportunity to work alongside more experienced designers. In addition, many of those who never made the transition to ‘expert’, would, nevertheless, still have the skills to build an accurate, complete, auditable, if unimaginative, conceptual schema.

### ***Self-appraisal***

Even when the full INTECoM framework is not taught, the use of the NaLER technique alone could provide a useful adjunct to the traditional techniques. NaLER allows student modellers to appraise their own work more critically, and thus begins to address the problem of ‘outcome irrelevant learning’ noted by Batra and Antony (1994). Use of

this technique should improve the students' ability to focus on what is actually recorded in an E-R/R model, in terms of information content, which would in turn improve their understanding of certain syntactical conventions of E-R/R modelling. The need to investigate the semantics of the model in detail, particularly the construction of examples, also encourages them to identify problems relating to inappropriate normalisation levels and incorrect designation of primary key attributes.

Initial experience with teaching NaLER has suggested to the researcher that students find little difficulty in learning or using the technique. Indeed, some remarked that it provided a structure for which they had intuitively recognised a need (Atkins & Patrick, 1998). Perhaps the most encouraging observation was that like any useful tool, it began to be used as a method of choice for solving problems other than that for which it was designed. For example, given the problem of deciding which of two models better represented a scenario description, some students developed a NaLER description for both models and compared them directly via these descriptions. A few students unknowingly followed the first step of the NIAM-CSDP, by creating a set of elementary fact sentences from the scenario description itself and which they then used as a benchmark against which to compare the descriptions derived from the models. They thus innovatively found a way to compare each NaLER description to a formal expression of the required semantics of the UoD which made it fairly straightforward to determine which 'facts' were usefully represented, which were missing or unclear, and which had been introduced by the modeller. This use of this technique has subsequently proved useful to the researcher in assessing data models constructed for assignments.

The use of NaLER encouraged students to assess the validity of the information that they were creating in their model and to be more precise in their use and interpretation of syntax. It has thus gone some way in encouraging the students to abstract meaning from the models rather than projecting meaning on to them. It also encouraged healthy questioning among students, for example, querying why a particular element had been designated as an entity rather than an attribute, or what was the basis on which categorisation had been decided.

## **Summary**

The adoption of the INTECoM approach to database development would necessitate a reappraisal of the skills that are required at different stages of the conceptual modelling process. In practice, this could result in an increase in the number of professionals able to successfully undertake data analysis and it is conceivable that data analysts could be recruited from the user base rather than from IS personnel. Data design would continue to be a highly skilled, technically oriented problem-solving activity in which experience and past achievement were essential ingredients. The emphasis of the training given to IS data developers would move away from attempting to teach a descriptive design technique and concentrate instead on providing them with a solid introduction to data analysis thereby laying a useful foundation from which to build in their professional careers.



# 15 Conclusion

*“True analysis always begins with the unknown...It is the job of the analyst to research the facts, exercise critical judgement and impose order and structure where none existed...(but)...there is no way to completely eliminate uncertainty from the analytical process” (Flavin, 1981 p.10).*

This study has investigated a number of issues within the area of conceptual data modelling and highlighted some of the confusion and contradictions that currently exist. Over the last twenty years, other researchers have undertaken similar exercises but many of the problems associated with these issues persist. In particular, academic wisdom and practitioner behaviour seem to have developed along very different lines and although there would appear to be a consensus on the importance of conceptual data modelling within IS development, recent studies have suggested that a large majority of organisations do not undertake it.

Even the very concept of a ‘conceptual data model’ is contentious. There are numerous definitions and although the differences between them may sometimes appear subtle, it is clear that there is no consensus on either the purpose, or the format of such a model. It does seem that, while lip service is paid to its use as a tool of communication, a conceptual data model is almost always constructed with the eventual database implementation very firmly in mind. As this implementation is, almost exclusively, relational there is a strong tendency to construct conceptual models which are heavily biased towards relational constructs. The greatest danger in this, aside from the loss of meta-data independence, arises from its unrecognised nature.

Based on a variety of evidence, the existence of a largely unacknowledged E-R/R hybrid model has been purported in this study. It has been suggested that, although a number of academic researchers continue to claim that Chen’s (1976) E-R Model is very widely used in practice, it is in fact the Relational Model, diagrammatically represented with E-

R notation, that is, in fact, the most ubiquitous. As this hybrid has become increasingly used for data model representations, at all three stages of the database design process, i.e. conceptual, logical and physical, it has exercised a significant effect on, among other things, the processes involved in constructing these models.

This study has brought together a number of criticisms that have been made of the E-R method; criticisms, which it has suggested, should be, more appropriately, directed at the E-R/R hybrid. These include claims that E-R Models are not that understandable or intuitive for non-specialists, not that well understood by practitioners, difficult to teach and difficult to evaluate for quality. Nevertheless, the E-R/R approach offers a powerful design technique, when used by skilled practitioners, and the hybrid Model provides a very suitable datalogical representation for relational data structures.

NIAM-CSDP, an alternative, less-widely used, conceptual modelling method has also been investigated. This study proposes that certain elements of this approach, particularly the provision of a prescriptive technique for establishing the data content of a UoD and the use of examples to verify the validity of the fact types, will facilitate novice learning and improve the analytical base from which data structures can be designed. In addition the use of natural language throughout the modelling process would make it more accessible to users. It has been observed that the NIAM-CSDP has not been shown to be any easier to use than its rival, and the results achieved by novices have not been proven to be any better. It has also been recognised that the prescriptive nature of the NIAM-CSDP, while a strength in some ways, could act as a constraint on the creativity required to find, apt and ingenious, solutions to difficult problems.

The major impact of the E-R/Relational hybrid on the data modelling process has been to increase the fuzziness of the boundary between data analysis and data design. The utilisation of the hybrid as the only tool for both stages has not only brought into question the effectiveness of the technique itself, but also served to conceal the essential differences between the two activities.<sup>1</sup> Consequently, this study also considered the characteristics of the two stages and suggested that it would be more productive to see

---

<sup>1</sup> However, in some methodologies, the phase of requirements elicitation becomes the first analysis phase and all conceptual modelling is seen as part of the design stage (e.g. Ram, 1995).

the E-R/R approach and the NIAM-CSDP as complementary rather than competitive. There seems good reason to suppose that a method, such as NIAM-CSDP, which employs a wide use of natural language would be useful in several infological ways but that there were other, datalogical, needs which were better met by elements of the existing E-R/R approach.

A tension certainly exists between the prescriptive and descriptive approach to conceptual data modelling as well as between the infological and datalogical roles such models are intended to play. This study has demonstrated that while there is a clear need for tools that assist both the analysis and the design stages, it is probably unreasonable to expect one tool to do both effectively. The question of how users, or modellers, think of the data, e.g. as objects, entities or something quite different (Weber, 1996), is perhaps less important than the fact that they generally communicate about them in natural language. Therefore, it seems reasonable to suggest that a framework, which provides for an analysis tool, that utilises natural language, to be integrated with a design tool, which both facilitates creative thought and can easily represent implementable data structures, will be both useful and powerful.

INTECoM is such a framework. By combining the prescriptive, analytical approach of the NIAM-CSDP with the essentially creative elements of E-R/R modelling, it utilises the strengths of both approaches while compensating for some of their weaknesses. Its own strength lies, partly in the fact that it does not require the introduction of a large number of new skills, methods or techniques and partly that it encompasses an already recognised framework for database development. While the use of INTECoM may necessitate a new attitude to the two techniques and how they fit into the overall development cycle, with the exception of NaLER, the techniques themselves are extensively documented and have been widely trialled. Although, experienced practitioners may find the framework overly constraining, the adoption of INTECoM could have a number of beneficial implications for the less experienced modellers, educators and those organisations that do not currently make sophisticated, if any, use of conceptual data modelling. In addition, the NaLER technique, even on its own, offers a pragmatic and easily learned method of 'reading' a model for those who have no requirement or interest in learning to construct one.

This study has brought together academic research, practitioner behaviour and educational experience from which to infer a theoretical framework. While all the major elements of the framework have been tested empirically or by prolonged use, INTECoM, itself, and its perceived benefits are still largely hypothetical. It has not been rigorously tested nor has it been taught as a formal method. The NaLER technique has been taught to two groups of final year undergraduate Information Systems students, but the evidence of its success is restricted to a small number of people and is anecdotal. However, there has been considerable interest from the educators with whom it has so far been discussed. The lack of an integrated CASE tool to support the INTECoM framework is a significant obstacle to its adoption by the wider commercial IS industry.

### **Future Work**

There are a number of fruitful areas of research that could be developed from this study. There is a paucity of research focussed on the area of data modelling practice (Srinivasan and Te'eni, 1995). A more detailed investigation into organisations' expectations of conceptual data modelling would provide a clearer foundation of the practical requirements of the activity, as would information on the behaviour of expert modellers in the areas of data analysis and data design. A more thorough study of the characteristics of these two stages of database development would assist in deciding whether the strengths of the E-R/R and NIAM-CSDP approaches would fit together, as usefully as this study has hypothesised and the worked example suggested. Further study is required into the ways in which graphical representations are perceived both by users and modellers and whether a natural language interpretation is the most useful form of interface that can be provided for non-technical users.

It would also be very useful to have a better understanding of the processes whereby novice modellers learn the skills of data analysis and data design. This would help in evaluating whether the use of a prescriptive method would reduce the learning curve and increase both their self-confidence and their effectiveness. There needs to be further discussion on the desirability, or not, of a prescriptive method for conceptual modelling and investigation of the contention that any modelling facility which uses the entity/attribute constructs is, by its very nature, precluded from developing a prescriptive approach. This study has suggested that the NIAM-CSDP could be useable as an

analysis technique by those with no specific IS skills and this contention also needs to be investigated more fully.

INTECoM as a framework needs to be the subject of a comprehensive study preferably undertaken on a suitably sized project within an organisation that does not currently use a sophisticated database development process. There does not appear to be any research comparing the ease of learning or the quality of models, produced by students, following E-R/R and NIAM training. It would be interesting to teach the INTECoM framework to undergraduate IS students. The new techniques involved would be an extension to the current skill set rather than an alternative and in this way, the impact of teaching a natural language analysis method on the students' ability to build successful and appropriate relational database structures, could be assessed.

The specification of a suitable integrated CASE tool would be valuable, as would either the construction of such a tool or the customisation of an existing one. This would also require an investigation of how INTECoM could best be integrated with other aspects of the information systems development life cycle. Another possibility would be to investigate the possibility of automating the production of a full set of example sentences from an E-R/R hybrid model, as it is held within current CASE tools, using the principles of the NaLER technique.

Finally, it is recognised that both INTECoM and NaLER have been described within the context of the E-R/R models that are the usual output of modelling in the commercial environment. However, object-oriented techniques are increasingly being used for systems analysis even where there is still a requirement to design a relational database; and the problems of translating these data requirements into an appropriate relational design are significant. There needs to be a much closer examination of the different elements inherent in object-oriented analysis and their applicability to the needs of data analysis and design. From this it could be determined whether they represent more appropriate alternative techniques and whether they could, in turn, be used to enhance the INTECoM method. While ORM and ER/R techniques have provided the means of instantiating the initial INTECoM framework, it is not these techniques themselves that are the essential ingredients but the 'ways of working' that they represent. The real contribution of the INTECoM framework is its emphasis, on making a clear distinction

between data analysis and design, improving user accessibility and facilitating quality auditing.

In keeping with Flavin's (1981) observations that head this chapter, this research began as a journey into the unknown. Facts were researched and critical judgement exercised. The desire to impose order and structure led to the visualisation of a framework that would assist this quest. However, in keeping with the content of the research, the actualisation of the framework, required creativity, innovation and a large measure of design. Just as there is ultimately, no definitive conceptual data model, so there is no definitive framework for creating one. Refinements, improvements, even, perhaps, better alternatives to INTECoM, are thus expected and welcomed. Indeed, if identifying the need for such a framework and constructing this prototype, encourages such a response, then the journey, undoubtedly, will have been worthwhile.

## References



- AMER, T.S. (1993): Entity-Relationship and Relational Database Modeling Representations for the Audit Review of Accounting Applications: An Experimental Examination of Effectiveness. *Journal of Information Systems*. 7(1): 1-15.
- ANSI (AMERICAN NATIONAL STANDARDS INSTITUTE) (1975): ANSI/X3/SPARC Study Group on Data Base Management Systems: Interim Report, *ACM SIGMOD Bulletin*:7(2).
- ATKINS, C.F. (1996): Prescription or Description: Some Observations on the Conceptual Modelling Process, in PURVIS, M. (ed.), *Proceedings of Software Engineering: Education and Practice Conference*, Dunedin New Zealand, January: 34-41.
- ATKINS, C.F. and PATRICK, J.D. (1998): NaLER: A Natural Language Method for Interpreting E-R Models, in PURVIS, M. (ed.), *Proceedings of Software Engineering: Education and Practice Conference*, Dunedin New Zealand, January: 2-9.
- AURISA (1991): Anonymous workshop handout, *Proceedings of the 19<sup>th</sup> Australian Conference on Urban and Regional Information Systems*, Wellington, New Zealand: 9-19.
- AVISON, D.E. and FITZGERALD G. (1995): *Information Systems Development: methodologies, techniques and tools (2<sup>nd</sup> Edition)*, McGraw-Hill, Maidenhead.
- AVISON, D.E. and WOOD-HARPER, A.T. (1990): *Multiview: An exploration in Information Systems Development*, McGraw-Hill, Maidenhead.
- AVISON, D.E. (1992): *Information Systems Development, A Database Approach (2nd edition)*, Blackwell Scientific Publications, Oxford.
- BACHMAN, C. (1969): The data structure diagrams, *Bulletin of ACM SIGFIDET* 1(2).
- BARDEN, J. (1994): Conceptual Models to the Desktop: Methods/Tools for Client/Server, *Proceedings of the 2<sup>nd</sup> NIAM-ISDM Conference*, Albuquerque, USA: 1-14.
- BARKER, R. (1990): *CASE\*Method: Entity Relationship Modelling*, Addison Wesley.
- BATINI, C., CERI, S. and NAVATHE, S. (1992): *Conceptual Database Design: an Entity Relationship Approach*, Benjamin Cummings, Redwood City, California.
- BATRA, D and DAVIES, J.G. (1992): Conceptual Data Modelling in Database Design: similarities and differences between novice and expert designers, *International Journal of Man-Machine Studies*, 37: 82-101.
- BATRA, D. and ANTONY S.R. (1994): Novice errors in conceptual database design, *European Journal of Information Systems*, 3(1): 57-69.

- BATRA, D. and MARAKAS G.M. (1995): Conceptual Data Modelling in Theory and Practice, *European Journal of Information Systems*, 4: 185-193.
- BATRA, D. and SEIN, MAUNG K. (1994): Improving conceptual database design through feedback, *International Journal of Human-Computer Studies*, 40: 653-676.
- BATRA, D. and SRINIVASAN, A. (1992): A review and analysis of the usability of data management environments, *International Journal of Man-Machine Studies*, 36: 395-417.
- BATRA, D. and ZANAKIS, S.H. (1994): A conceptual database design approach based on rules and heuristics, *European Journal of Information Systems*, 3(3): 228-239.
- BATRA, D., HOFFER, J.A. and BOSTROM, R.P. (1990): Comparing Representations with Relational and EER Models, *Communications of the ACM*, 33(2): 126-139.
- BENYON, D. (1997): *Information and Data Modelling (2<sup>nd</sup> Edition)*, McGraw-Hill, London.
- BENYON-DAVIES, P. (1992a): Entity models to object models: object oriented analysis and database design, *Information and Software Technology*, 34(4): 255-262.
- BENYON-DAVIES, P. (1992b): The realities of database design: an essay on the sociology, semiology and pedagogy of database work, *Journal of Information Systems*, 2: 207-220.
- BENYON-DAVIES, P. (1996): *Database Systems*, Macmillan Press, London.
- BILLER, H. and NEUHOLD, E. (1977): Concepts for the Conceptual Schema, in NIJSSEN, G. (ed.), *Architecture and Models in DataBase Management Systems*, North-Holland, Amsterdam.
- BILLER, H., and NEUHOLD, E. (1978): Semantics of Data Bases: The Semantics of Data Models, *Information Systems* 3: 11-30.
- BLAHA, M.R., PREMERLANI, W.J. and RUMBAUGH, J.E. (1988): Relational Database Design using an Object-Oriented Methodology, *Communications of the ACM*, 31(4): 414-427.
- BLAHA, M.R. and PREMERLANI, W.J. (1997): *Object Oriented Modeling and Design for Database Applications*, Prentice Hall
- BOCK, D.B. and RYAN, T. (1993): Accuracy in Modeling with Extended Entity Relationship and Object Oriented Data Models, *Journal of Database Management*, 4(4): 30-39.
- BOLAND, R.J. (1987): The In-formation of Information Systems, in: BOLAND, R.J. and HIRSCHHEIM, R. (eds.). *Critical Issues in Information Systems*. Wiley, Chichester.

- BOLAND, R.J. (1991): Information Systems Use as a Hermeneutic Process, in NISSEN, H-E. KLEIN, H.K. and HIRSCHHEIM R.A. (eds.), *Information Systems Research: Contemporary Approaches and Emergent Traditions*, North-Holland, Amsterdam.:439-464.
- BOOCH, G. (1991): *Object Oriented Design: With Applications*, Benjamin Cummings, Menlo Park California.
- BOOCH, G. (1995): *Object Solutions*, Addison-Wesley, Redwood City, California.
- BOUZEGHOUB, M. and GARDARIN, G. (1984): The design of an expert system for database design, in GARDARIN, G. and GELENBE, E. (eds.), *New Applications of Databases*, Academic Press, London (quoted in Ram, 1995).
- BRODIE, M.L., MYLOPOULOS, J. and SCHMIDT, J.W. (eds.)(1984): *On Conceptual Modelling*, Springer-Verlag, New York.
- BRONTS, G., BROUWER, S.J., MARTENS, C.L.J. and PROPER, H.A. (1995): A Unifying Object Role Modelling Theory, *Information Systems*, 20(3): 213-235.
- BROWN, L. (ed.) (1993): *The New Shorter Oxford English Dictionary*, Clarendon Press Oxford.
- BUBENKO, J.A. (1986): Information Systems Methodologies: A research view, in OLLE, T.W., SOL, H.G., VERRIJN-STUART, A.A. (eds.) *Information Systems Design Methodologies: Improving the Practice*, North-Holland, Amsterdam.
- BUCKINGHAM Shum, S. and HAMMOND, N. (1994): Argumentation-based Design Rationale: what use at what cost, *International Journal of Human-Computer Studies*, 40: 603-652.
- BURCH, J.G. (1992): *Systems Analysis, Design and Implementation*, boyd and fraser publishing company, Boston.
- BURMEISTER, O.K. (1995): Evaluating the Factors that Facilitate Deep Understanding of Data Analysis, *Australian Journal of Information Systems*, 3(1): 2-13.
- BURRELL, G., and MORGAN, G. (1979): *Sociological Paradigms and Organisational Analysis*, Heinemann Educational Books. London (quoted in de Carteret and Vidgen, 1995).
- CALWAY, B.A. and SYKES J. (1995): A Grammatical Conversion of Descriptive Narrative - An Application of Discourse Analysis in Conceptual Modelling, *Australian Journal of Information Systems*, 3(2): 10-19.
- CAMPBELL, D. (1992): Entity-Relationship Modeling; One Style Suits All? *Database Summer*: 12-18.
- CARROLL, J.M. (ed.)(1995): *Scenario-Based Design*, John Wiley and Sons, New York.

- CARROLL, LEWIS (1871): *Through the Looking Glass*, reprinted 1965 in *The Works of Lewis Carroll*, Paul Hamlyn, London.
- CATTELL, R.G.G. (1994): *Object Data Management: Object-Oriented and Extended Relational Database Systems (revised edition)*, Addison Wesley, Reading, MA.
- CCTA (1994): *Corporate Data Modelling*, Her Majesty's Stationery Office, London.
- CERPA, N. (1995): Pre-Physical data base design heuristics, *Information and Management* 28: 351-359.
- CHECKLAND, P. and SHOLES, J. (1990): *Soft Systems Methodology in Action*, Wiley, Chichester.
- CHEN, P.P. (1976): The Entity-Relationship Model - Toward a Unified View of Data, *ACM Transactions on Database Systems*, 1(1): 9-36.
- CHEN, P.P. (1983): English Sentence Structure and Entity-Relationship Diagrams, *Information Sciences* 29(2): 127-149.
- CIBORRA, C.U. (1997): Crisis and Foundations: An Inquiry into the Nature and Limits of Models and Methods in the IS Discipline, *Proceedings of the 5<sup>th</sup> European Conference of Information Systems*, Cork, Ireland: 1549-1560.
- COAD, P. and YOURDON E. (1991): *Object Oriented Analysis*, Prentice Hall.
- CODASYL (1971): Report of the CODASYL Database Task Group, *Communications of the ACM* April (quoted in Navathe, 1992).
- CODD, E.F. (1970): A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM* 13 (6): 377-387.
- CODD, E.F. (1990): *The Relational Model for Database Management Version 2*, Addison-Wesley, Reading, MA.
- COLLINGNON, M.A. and VAN DER WEIDE, T.P. (1994) An Information Analysis Method Based on PSM. , *Proceedings of the 2<sup>nd</sup> NIAM-ISDM Conference*, Albuquerque, New Mexico:
- CONNOLLY, T.M., BEGG, C.E and STRACHAN, A.D. (1995): *Database Systems*, Addison Wesley, Wokingham, England.
- CREASY, P. (1989): ENIAM: a more complete conceptual schema language. *Proceedings of the 15<sup>th</sup> Conference on Very Large Databases*, Amsterdam, September (quoted in Laender and Flynn 1994).
- CREASY, P. and MOULIN, B. (1992): Adding Semantics to Semantic Data Models, in (Nagle *et al.*, 1992).

- DARKE, P. and SHANKS, G. (1994a): Viewpoint Development for Requirements Definition: An analysis of concepts, issues and approaches, *Dept of Information Systems, Monash University Working Paper Series 21/94* Melbourne, Australia.
- DARKE, P and SHANKS, G. (1994b): Defining Systems Requirements: A critical assessment of the NIAM Conceptual Schema Design Procedure, *Dept of Information Systems, Monash University Working Paper Series 8/94* Melbourne, Australia.
- DARKE, P. and SHANKS, G. (1995a): Viewpoint Development for Requirements Definition: Towards a Conceptual Framework, *Dept of Information Systems, Monash University Working Paper Series 9/95* Melbourne, Australia.
- DARKE, P. and SHANKS, G. (1995b): The use of Viewpoint Development to Enhance Requirements Definition with the NIAM Conceptual Schema Design Procedure, *Dept of Information Systems, Monash University Working Paper Series 12/95* Melbourne, Australia.
- DARKE, P. and SHANKS, G. (1995c): Defining Systems Requirements: A critical assessment of the NIAM conceptual schema, *Australian Journal of Information Systems*, 2(2): 50-62
- DATE, C.J. (1986): *Relational Database selected writings*, Addison-Wesley, Reading, Massachusetts.
- DATE, C.J. (1995): *An Introduction to Database Systems (6<sup>th</sup> Edition)*, Addison-Wesley, Reading, Massachusetts.
- de CARTERET, C. and VIDGEN, R. (1995): *Data Modelling for Information Systems*, Pitman Publishing, London.
- DURDING, B.M., BECKER, C.A. and GOULD, J.D. (1977): Data Organisation, *Human Factors*, and 19(1): 1-14.
- EAGLESTONE, B. and RIDLEY, M. (1998): *Object Databases: An Introduction*, London, McGraw-Hill.
- EDEN, P. (1996): A Method for Conceptual Data Analysis using Entity-Relationship Diagrams and Functional Dependency Diagrams, *Proceedings of Software Engineering: Education and Practice Conference*, Dunedin New Zealand, January: 42-48.
- ELMASRI, R., WEELDRYER, J. and HEVNER, A. (1985): The Category Concept: An extension to the Entity-Relationship Model, *Data and Knowledge Engineering*, 1: 75-116.
- ELMASRI, R. and NAVATHE, S.B. (1989): *Fundamentals of Database Systems*, Benjamin Cummings.
- EVA, M. (1994): *SSADM version 4: a users guide (2<sup>nd</sup> edition)*, McGraw-Hill, England.

- EVEREST, G. (1994): Experiences in teaching NIAM/OR modelling, *Proceedings of 2nd NIAM-ISDM conference*, Albuquerque, New Mexico, USA.
- FALKENBERG, E.D. (1976): Concepts for modelling information, in NIJSSSEN G.M. (ed.) *Modelling in Database Management Systems*, North-Holland Amsterdam, (quoted in Nijssen and Halpin, 1989).
- FILLMORE, C.J. (1968): The case for case, in BACHE E. and HARMS, R.T *Universals in Linguistic Theory*, Holt, Rhinehart and Winston, New York: 1-88 (quoted in Halpin, 1995).
- FINKELSTEIN, C. (1989): *An Introduction to Information Engineering: From Strategy Planning to Information Systems*, Addison Wesley, Sydney.
- FIRNS, P.G. (1990): Determining a Useful Balance between Understandability and Rigour in Data Modelling, *New Zealand Journal of Computing* 2(1): 13-21.
- FIRNS, P.G. (1993): *A Data Modelling and Schema Generation Tool for Use in System Development Courses*, Presentation, Department of Information Sciences, University of Otago, Dunedin, New Zealand: 111-119.
- FLAVIN, M. (1981): *Fundamental Concepts of Information Modelling*, Yourdon, New York.
- FLYNN, D. (1998): *Information Systems Requirements: Determination and Analysis (2<sup>nd</sup> edition)*, McGraw-Hill, London.
- GALLIERS, R.D. (1993): Research Issues in Information Systems, *Journal of Information Technology*, 8(2): 92-98.
- GERROLD, D. (1988): *When H.A.R.L.I.E. was One*, Bantam Books, USA.
- GIBSON, M.L. and HUGHES C.T. (1994): *Systems Analysis and Design. A Comprehensive Methodology with CASE*, boyd and fraser publishing company, Massachusetts.
- GOLDSTEIN, R.C. and STOREY, V.C. (1990): Some findings of the intuitiveness of entity-relationship constructs, in LOCHOVSKY, F.H., (ed.), *Entity-Relationship Approach to Database Design and Querying*, Elsevier Science Amsterdam.
- GREEN, P.F. (1997): Use of Information Systems Analysis and Design (ISAD) Grammars in Combination in Upper Case Tools - An Ontological Evaluation, *Proceedings of the Second CAiSE/IFIP 8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD)*, Barcelona, Spain.
- GRIMES, S. (1998): Modeling Object/Relational Databases, *DBMS*, April: 11(4): 51-55.
- HALPIN, T.A. and ORLOWSKA, M.E. (1992): Fact-oriented modelling for data analysis, *Journal of Information Systems* 2.

- HALPIN, T.A. (1993a): An overview of Object-Role Modelling, in PERSCHKE, S. and LICZBANSKI, M. (eds.) *Access for Windows Power Programming*, Que Corporation, Carmel, IN, USA.
- HALPIN, T.A. (1993b): What is an elementary fact? *Proceedings of 1st NIAM-ISDM Conference*, Utrecht, The Netherlands.
- HALPIN, T.A. (1995): *Conceptual Schema and Relational Database Design (2<sup>nd</sup> edition)*, Prentice Hall, Australia.
- HAMMER, M. and MCLEOD, D. (1981): Database Description with SDM: A Semantic Database Model, *ACM Transactions on Database Systems*, 6(3): .351-386.
- HANKS P. (ed.)(1979): *Collins Dictionary of the English Language*, Collins.
- HAWRYSZKIEWYCZ, I. (1987): User Experience with the E-R Approach, Report on the Panel Session, Prepared by Gary Schuldt, in SPACCAPIETRA, S. (ed.), *Entity-Relationship Approach*, Elsevier Science Publishers B.V., North-Holland, Amsterdam: 465-472.
- HAWRYSZKIEWYCZ, I. (1997): *Introduction to Systems Analysis and Design (4<sup>th</sup> edition)*, Prentice Hall, Sydney.
- HERBET, F. (1972): *Dune*, New English Library, Sevenoaks, Kent.
- HIRSCHHEIM, R., KLEIN, H.K. and LYYTINEN, K. (1995): *Information Systems Development and Data Modelling, Conceptual and Philosophical Foundations*, Cambridge University Press, Cambridge.
- HITCHMAN, S. (1995): Practitioner perceptions on the use of some semantic concepts in the entity-relationship model, *European Journal of Information Systems*, 4: 31-40.
- HOFSTADTER, D.R. (1979): *Gödel, Escher, Bach: An Eternal Golden Braid*, The Harvester Press, UK.
- HOWE, D.R. (1989): *Data Analysis for Database Design (2<sup>nd</sup> edition)*, Edward Arnold, and London.
- HUTCHINS, E.L., HOLLAN, J.D. and NORMAN, D.A. (1985): Direct Manipulation Interfaces, *Human Computer Interaction* 1: 311-338.
- HUFF, H.W. (1992): The Relational Model contra Entity-Relationship? *SIGMOD Record*, 21(3): 33-34.
- HULL, R. and KING, R. (1987): Semantic database modeling: Survey, applications and research issues, *ACM Computing Surveys*, 19(3): 201-260.
- IFIP' ICC (1966): *Vocabulary of Information Processing*, North Holland, Amsterdam (quoted in Olle, 1993).

- IS 10027(1993): Reference Model of Data Management, (quoted in Olle, 1993).
- ISO/TC97/SC5/WG3 Report (March 1982) In van Griethuyzen (1983).
- JACOBSON, I., CHRISTERSON, M., JONSSON, P. and OVERGAARD, G. (1992): *Object-Oriented Software Engineering: A Use Case Driven Approach*, Addison-Wesley, Wokingham.
- JARVENPAA, S.L and MACHESKY, J.J. (1989): Data Analysis and Learning: an experimental study of data modelling tools, *International Journal of Man-Machine Studies*, 31:367-391.
- JIH, W.K., BRADBARD, D.A., SNYDER, C.A. and THOMPSON, N.G.A., (1989): The effects of relational and entity-relationship data models on query performance of end users, *International Journal of Man-Machine Studies*, 31: 257-267.
- JUHN, S. and NAUMANN, J.D. (1985): The effectiveness of data representation characteristics on user validation, *Proceedings of the Sixth International Conference on Information Systems*. Indianapolis: 212-226.
- KENT, W. (1978): *Data and Reality*, North-Holland, Amsterdam.
- KENT, W. (1979): Limitations of Record-based Information Models, *ACM Transactions on Database Systems*, 4(1): 107-131.
- KENT, W. (1981): Consequences of Assuming a Universal Relation, *ACM TODS*, 6(4) (quoted in Date, 1995).
- KENT, W. (1983): Fact-based Analysis and Design, in DAVIES, C.G., JAJODIA, S., NG, P.A. and YEH, R.T. (eds.), *Proceedings of the Entity-Relationship Approach to Software Engineering*, Elsevier Science Publishers, Amsterdam: 3-53.
- KEPNER, C.H. (1996): Calling all thinkers, *H R Focus*, 73(10): 3.
- KESH, S. (1995): Evaluating the quality of entity relationship models, *Information and Software Technology*, 37(12): 681-689.
- KEUFFEL, W. (1996): Battle of the Modeling Techniques, *DBMS*, August: 83-84,86,97
- KIM, Y-G. and MARCH S.T. (1995): Comparing Data Modeling Formalisms, *Communications of the ACM*. 38(6): 103-115.
- KINN, S. and SULLIVAN, F. (1994): Good relations, *The British Journal of Healthcare Computing and Information Management*, 11(4) May: 21-23.
- KLEIN, H. and HIRSCHHEIM, R.A. (1987): A Comparative Framework of Data Modelling Paradigms and Approaches, *The Computer Journal*, 30(1): 8-15.
- KLEIN, H. and MYERS, M. (1999): A Set Of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems, *MIS Quarterly*, and 23(1): 67-93.

- KONSYNSKI, B.R. (1979): *Data Base Driven Systems*, University of Arizona, (quoted in Hitchman, 1995)
- KORZYBSKI, A. (1941): *Science and Sanity*, Science Press, New York (quoted in Benyon-Davies, 1996).
- KOSKO, B. (1993): *Fuzzy Thinking: the new science of fuzzy logic*, Harper Collins, London.
- KROENKE, D.M. (1992): *Database Processing: Fundamentals, Design, Implementation (4<sup>th</sup> edition)*, Maxwell Macmillan International.
- KROGSTIE, J., LINDLAND O.I and SINDRE, G. (1995): Towards a deeper understanding of Quality in Requirements Engineering, *Proceedings of 7th CAiSE, Jyväskylä, Finland*. June: unnumbered.
- LAENDER, A.H.F. and FLYNN, D.J. (1994): A semantic comparison of the modelling capabilities of the E-R and NIAM models, *Lecture notes in Computer Science* No 823 Springer-Verlag: 242-256.
- LAKOFF, G. (1987): *Women, Fire and Dangerous Things*, The University of Chicago Press.
- LANGFORS, B. (1963): Some Approaches to the Theory of Information Systems, *BIT* 3: 229-254 (quoted in Hirschheim *et al.*, 1995).
- LARMAN, C. (1998): *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design*, Prentice Hall, Upper Saddle River, New Jersey.
- LEUNG, C.M.R. and NIJSSEN, G. M. (1988): Relational Database Design using the NIAM Conceptual Schema, *Information Systems*, 13(2): 219-227.
- LINDLAND, O.I., SINDRE, G and SØLVBERG, A. (1994): Understanding Quality in Conceptual Modelling, *IEEE Software*, March: 42-49.
- LOOMIS, M.E.S. (1987): *The Database Book*, Macmillan, New York.
- LOOSLEY, C. and GANE, C. (1990): Information Systems Modeling Part 1: The object of processing is data, *InfoDB*, 4(4) this version reprinted for Bachman: 1-14.
- LYYTINEN, K.J. and KLEIN, H.K. (1985): The Critical Theory Of Jurgen Habermas As A Basis For A Theory Of Information Systems, in MUMFORD, E *et al.* (eds.), *Research Methods in Information Systems*, Elsevier Science Publishers B.V. (North-Holland): 219-235.
- MACDONELL, S.G. (1994): Software Development, CASE tools and 4GLs - A Survey of New Zealand Usage. Part 1: 750 New Zealand Organisations, *New Zealand Journal of Computing*, 5(1): 23-33.
- MACEACHREN, A.E. (1995): *How Maps Work*, The Guildford Press, New York.

- MANNILA, H and RÄIHÄ, K.-J. (1992): *The Design of Relational Databases*, Addison Wesley.
- MARCHE, S. (1993): Measuring the stability of data models, *European Journal of Information Systems*, 2(1) 37-47.
- MARTIN, J. and ODELL, J. (1992): *Object-Oriented Analysis and Design*, Prentice-Hall, Englewood Cliffs.
- MARTIN, J. (1990): *Information Engineering. Book II: Planning and Analysis*, Prentice-Hall, New Jersey.
- MARTINDALE, V. (1997): *Personal Communication*, Eagle Star Insurance Group, Cheltenham, UK.
- MAYER, R.E. (1989): Models for Understanding, *Review of Educational Research*, 59(1): 43-64.
- MCFADDEN, F.R. and HOFFER, J.A. (1994): *Database Management*, Benjamin/Cummings, California.
- MÉTAIS E., MEUNIER, J.-N. and LEVREAU, G. (1993): Database Schema Design: A Perspective from Natural Language Techniques to Validation and View Integration, *Proceedings of the 12<sup>th</sup> International Conference on Entity-Relationship Approach*, Arlington, Texas, 190-205.
- MOODY, D. (1996): The Seven Habits of Highly Effective Data Modelers, *Database Programming and Design*, October: 57-64.
- MOODY, D. and OSIANLIS, A. (1996): Bringing Data Models to "Life": An Interactive Tool for Representing Entity Relationship Models, *Proceedings of the 7th Australian Conference on Information Systems (ACIS'96)*. University of Tasmania Australia: 497-508.
- MOODY, D. and SHANKS, G. (1994): What Makes a Good Data Model? Evaluating the Quality of Entity Relationship Models, *Dept of Information Systems, Monash University Working Paper Series 12/94*, Melbourne, Australia.
- MOODY, D. and SHANKS, G. (1998): What Makes a Good Data Model? A Framework for Evaluating and Improving the Quality of Entity Relationship Models, *The Australian Computer Journal*, 30(3): 97-110.
- NAGLE, T.E., NAGLE J.A., GERHOLZ, L.L. and EKLUND, P.W. (eds.)(1992): *Conceptual Structures: Current Research and Practice*, Ellis Horwood, New York.
- NAVATHE, S.B. (1992): Evolution of Data Modelling for Databases, *Communications of the ACM*, 35 (9): 112-123.
- NIJSSSEN, G.M. and HALPIN, T.A. (1989): *Conceptual Schema and Relational Database Design*, Prentice Hall.

- NIJSSEN, G. M. (1994): A General Analysis Procedure, *Proceedings of the 2<sup>nd</sup> NIAM-ISDM Conference*, Albuquerque, New Mexico USA: B1-B15.
- NORDBOTTEN, J.C. and CROSBY, M.E. (1996): Reading strategies for graphic models from an experiment in data model perception, *Proceedings of 5th International Conference on User Modeling*, Hawaii, USA, January: 43-48.
- OLDS, D. (1997): *Personal Communication*, IBM Int. (NZ).
- OLLE, T.W., HAGELSTEIN, J., MACDONALD, I.G., SOL, H.G., VAN ASSCHE, F.J.M. and VERRIJN-STUART, A.A. (1991): *Information Systems Methodologies: A Framework for Understanding* (2<sup>nd</sup> edition), Addison Wesley.
- OLLE, T.W. (1993): Data Modelling and Conceptual Modelling: A comparative analysis of functionality and roles, *Australian Journal of Information Systems*, 1(1): 46-57.
- PAGE JONES, M. (1988): *The Practical Guide to Structured System Design*, Prentice-Hall, New Jersey.
- PAVLIA, P.C., LIAO, C. and TO, P-L. (1992): The Impact of Conceptual Data Models on End-User Performance, *Journal of Database Management*, 3(4): 4-15.
- PECKHAM, J. and MARYANSKI, F. (1988): Semantic Data Models, *ACM Computing Surveys*, and 20(3): 153-189.
- PLETCH, A. (1989): Conceptual Modelling in the Classroom, *SIGMOD RECORD*, and 18(1): 74-80.
- POHL, K. (1994): The three dimensions of requirements engineering: A Framework and its applications, *Information Systems*, 19(3): 243-258.
- POST, G.V. (1999): *Database Management Systems: Designing and Building Business Applications*, Irwin/McGraw-Hill, Boston, USA.
- QUATRANI, T. (1998): *Visual Modeling with Rational Rose and UML*, Addison-Wesley, Reading Mass.
- RAM, S. (1995): Deriving Functional Dependencies from the Entity-Relationship Model, *Communications of the ACM*, 58(9): 95-107.
- RAYMOND, D.R., CANAS, A.J., TOMPA, F.W. and SAFAYENI F.R. (1989): Measuring the effectiveness of personal database structures, *International Journal of Man-Machine Studies* 31: 237-256.
- RUMBAUGH, J., BLAHA, M., PREMERLANI, W., EDDY, F. and LORENSON W. (1991): *Object-Oriented Modelling and Design*, Prentice-Hall International, New York.
- RICARDO, C. (1990): *Database Systems: Principles, Design and Implementation*, Macmillan, New York.

- ROBINSON, M. and BANNON, L. (1991): Questioning Representations, Proceedings of 2<sup>nd</sup> European Conference on Computer-Supported Cooperative Work, Amsterdam, The Netherlands, Spring (quoted in Darke and Shanks, 1995b).
- RYDER, M.R. (1993): *A CASE for Relational Modelling*, Unpublished thesis, Massey University, New Zealand.
- RYDER, M.R. (1996): *Facilitating Evolution in Relational Database Design: A procedure to evaluate and refine novice database designers' schemata*, Unpublished thesis, Massey University, New Zealand.
- SANDERS, G.L. (1995): *Data Modeling*, boyd and fraser, ITP Inc, Danvers, Massachusetts.
- SAPIR, E. (1931): Conceptual categories in Primitive Languages, *Science* (74), (quoted in Kent, 1978).
- SCHENCK, D.A. and WILSON P.R. (1994): *Information Modeling: The EXPRESS Way*, Oxford University Press, New York.
- SCHOUTEN, H. (1993): A Comparison of Conceptual Graphs with NIAM, *Proceedings of NIAM-ISDM Conference Utrecht*, The Netherlands: 1-29.
- SCHOUTEN, H. (1994): How to formally specify the NIAM information analysis method, *Proceedings of the 2<sup>nd</sup> NIAM-ISDM Conference*, Albuquerque, New Mexico: E1-E36.
- SHANKS, G. (1997): Conceptual Data Modelling: An empirical study of expert and novice data modellers, *Australian Journal of Information System*, 4(2): 63-73.
- SHANKS, G. and DARKE, P. (1996): Understanding and Evaluating the Quality of Conceptual Models: Linking Theory and Practice, *Dept. of Information Systems, Monash University Working Paper Series 7/96*, Melbourne, Australia.
- SHANKS, G. and DARKE, P. (1997): Using Explanation and Visualisation to Improve Understanding of Corporate Data Models, *Proceedings of 8<sup>th</sup> Australian Conference on Information Systems*, Tasmania, Australia.
- SHANKS, G., SIMSION G. and REMBACH, M. (1993): The Role of Experience in Conceptual Schema Design, *Dept. of Information Systems, Monash University Working Paper Series 2/93*, Melbourne, Australia.
- SHARP, J.K. (1993): A comparison of IDEF1X and NIAM-ISDM, *Proceedings of the NIAM-ISDM Conference*, Utrecht, The Netherlands: 1-13.
- SHARP, J.K. (1994): Adopting the Natural Language Information Analysis Methodology in your corporation, *Proceedings of the 2<sup>nd</sup> NIAM-ISDM Conference*, Albuquerque, New Mexico: D1-D17.
- SHAVE, M.J.R. (1981): Entities, functions and binary relations: steps to a conceptual schema, *The Computer Journal*, 24(1): 42-46.

- SHOVAL, P. and EVEN-CHAIME, M. (1987): Database schema design: An experimental comparison between normalisation and information analysis, *Database*, 18(3), Spring: 30-39.
- SHOVAL, P. and FRUMERMANN, I. (1994): OO and EER Conceptual Schemas: A comparison of User Comprehension, *Journal of Database Management*, 5(4): 28-38.
- SHOVAL, P. (1982): From Information Needs to Database Conceptual Design, *Proceedings of the Annual IPA Conference on Data Processing*, Jerusalem, Israel, pp.41-61.
- SHOVAL, P. (1985): Essential Information Structure Diagrams and Database Schema Design, *Information Systems*, 10(4): 417-423.
- SHOVAL, P. (1997): Experimental Comparisons of Entity-Relationship and Object-Oriented Data Models, *Australian Journal of Information Systems*, 4(2): 74-81.
- SIAU, K., WAND, Y. and BENBASAT I. (1995): A Psychological Study on the Use of Relationship Concept - Some Preliminary Findings, in ILVARI, J., LYYTINEN, K., ROSSI, M. (eds.), *Lecture Notes in Computer Science*, 932. Springer: 341-354.
- SIAU, K., WAND, Y. and BENBASAT I. (1996): When Parents Need Not Have Children - Cognitive Biases in Information Modeling, in CONSTANTOPOULOS, P., MYLOPOULOS and VASSILIOU, T, (eds.), *Lecture Notes in Computer Science*, 1080. Springer: 402-419.
- SIMSION, G. and SHANKS G. (1993): Choosing Entity Types - A Study of 51 Data Modellers, *Dept of Information Systems, Monash University Working Paper Series 17/93*, Melbourne, Australia.
- SIMSION, G., (1994): *Data Modelling Essentials Analysis, Design and Innovation*, Van Nostrand Reinhold, Boston.
- SMITH, J.M. and SMITH, D.C.P. (1977a): Database Abstractions: Aggregation and Generalisation, *ACM Transactions on Database Systems*, 2(2): 105-133.
- SMITH, J.M. and SMITH, D.C.P. (1977b): Database Abstractions: Aggregation, *Communications of the ACM*, 20(6): 405-413.
- SONG, I-Y. and FORBES, E.A. (1991): Schema conversion rules between EER and NIAM model, *Proceedings of the 10<sup>th</sup> International Conference on Entity-Relationship Approach*, San Mateo, California. October.
- SOWA, J.F. (1984): *Conceptual Structures: information processing in mind and machine*, Addison Wesley Reading Massachusetts.
- SOWA, J.F. (1991): Towards the Expressive Power of Natural Languages, in J.F. SOWA, J.F. (ed.), *Principles of Semantic Networks*, Morgan Kaufmann, San Mateo, CA.

- SRINIVASAN, A. and TE'ENI, D. (1995): Modeling as Constrained Problem Solving: An Empirical Study of the Data Modeling Process, *Management Science*, 41(3): 419-434
- STEINBERG, G., FALEY, R. and CHINN, S. (1994): Automatic Database Generation by Novice End-Users Using English Sentences, *Journal of Systems Management*, March: 10-15.
- STOREY, V. and GOLDSTEIN R. (1988): A Methodology for Creating User Views in Database Design, *ACM Transactions on Database Systems*. 13(3): 305-338.
- STOREY, V. and GOLDSTEIN R. (1993): Knowledge-Based Approaches to Database Design, *MIS Quarterly*, March: 25-46.
- SUTCLIFFE, A.G. and MAIDEN, N.A.M. (1992): Analysing the novice analyst: cognitive models in software engineering, *International Journal of Man-Machine Studies*, 36: 719-740.
- TEOREY, T.J. and FRY, J.P. (1982): *Design of Database Structures*, Prentice-Hall, Englewood Cliffs, New Jersey.
- TEOREY, T.J., YANG, D. and FRY, J.P. (1986): A Logical Design Methodology for Relational Databases using the Extended Entity-Relationship Model, *Computing Surveys*, 18(2) June: 197-222.
- TJOA, A.M. and BERGER, L. (1993): Transformation of Requirements Specifications expressed in Natural Language into an EER Model, *Proceedings of the 12<sup>th</sup> International Conference on Entity Relationship Approach*, Arlington, Texas: 206-217.
- TOLIS, C. (1996): Working with models in development work: Differences that hinder or facilitate, *Proceedings of the 7<sup>th</sup> Australian Conference on Information Systems*, (ACIS'96) University of Tasmania, Hobart, Tasmania, 2: 735-745.
- TSICHRITZIS, D.C. and LOCHOVSKY, F.H. (1982): *Data Models*, Prentice Hall, Englewood Cliffs, New Jersey.
- VADERA, S. and MEZIANE, F. (1994): From English to Formal Specifications, *The Computer Journal*, and 37(9): 753-763.
- van der LEK, H., BAKEMA, G.P. and ZWART, J.P.C. (1992): Unifying object types and fact types: a practically and didactically productive theory, translated from the Dutch by BERTHOFF, T and ZWART, J.P.C., De inificatie van objecttypen en feittypen, een praktisch en didactisch vruchtbare theorie, *Informatie*, 34(5): 279-295,
- van GRIETHUYZEN, J.J. (ed.), (1983): *Concepts and Terminology for the Conceptual Schema and the Information Base ISO/TC97/SC5-N695*, International Organization for Standardization, Geneva, Switzerland (quoted in Olle, 1993).

- VERYARD, R. (1984): *Pragmatic Data Analysis*, Blackwell Scientific Publication Oxford.
- VERYARD, R. (1992): *Information Modelling: Practical Guidance*, Prentice Hall, Englewood Cliffs, New Jersey.
- VERYARD, R. (1994): *Information co-ordination: The management of Information Models, Systems and Organizations*, Prentice Hall Englewood Cliffs, New Jersey
- WALSHAM, G. (1993): *Interpreting Information Systems in Organizations*, Wiley, Chichester, England
- WAND, Y. and WEBER, R. (1993): On the ontological expressiveness of information systems analysis and design grammars, *Journal of Information Systems*, 3(4): 217-237.
- WAND, Y. and WEBER, R. (1995): On the deep structure of information systems, *Information Systems Journal*, 5: 203-223.
- WATSON, R.T. (1996) *Data Management: An Organizational Perspective*, John Wiley, New York.
- WAY, E. C. (1991): *Knowledge Representation and metaphor*, Kluwer Academic Publishers, Dordrecht, The Netherlands.
- WEBER, R. and ZHANG, Y., (1991) An ontological evaluation of NIAM's grammar for conceptual schema design, *Proceedings of the 12<sup>th</sup> International Conference on Information Systems*, New York: 75-82.
- WEBER, R., (1996): Are Attributes Entities? A study of Database Designers' Memory Structures, *Information Systems Research*, 7(2) June.
- WEBER, R. (1997): The Link between Data Modeling Approaches and Philosophical Assumptions: A Critique, *Proceedings of the Association of Information Systems Conference*, Indianapolis: 306-308.
- WONG CHEE-PUN, (1995): Fried Rice: A Recipe for Data Modeling and Design, *Database Programming and Design*, April: 41-44.
- YAO, S.B., NAVATHE, S.B., and WELDON, J.L. (1982): An integrative approach to database design, in *Database Design Techniques 1: Requirements and Logical Structures Proceedings* (quoted in Hitchman, 1995).
- YUNKER, K. (1993): The Dependency between Representation and Procedure, *Proceedings of the NIAM-ISDM Conference*, Utrecht, The Netherlands: 1-27.
- ZACHMAN, J. (1987): A Framework for Information Systems Architecture, *IBM Systems Journal* 26(3): 276-292.



# Glossary

<b>analysis</b>	In general terms, an activity which seeks to determine the elements or components of something complex and to discover the general principles underlying these concrete phenomena. In IS development it is the decomposition of problems into their component parts.
<b>analysis model</b>	A formal and precise specification of the users identified requirements for a system under development. Ideally, this record will be a specification of what the needs are, with no consideration of how they will be implemented physically.
<b>analysis stage</b>	In IS development, the stage where the user requirement is identified and specified, usually in a formal and precise record.
<b>CASE tool</b>	Computer Aided Systems (or Software) Engineering. A software package that provides computer support for key activities in the system development process.
<b>conceptual data model</b>	A model, or collection of models, that records the information requirements of a system with no consideration of the specific technology by which it will be implemented. Sometimes referred to as the 'logical model' of the system.  In this study, the conceptual data model is considered to encompass both the analysis and the paradigmatic design models.
<b>conceptual data modelling</b>	The process by which a conceptual data model is constructed.  In this study, this process includes both analysis and design activities.
<b>conceptual schema</b>	A name sometimes given to the design version of the conceptual data model, usually with an emphasis on the data structuring aspect of the model.
<b>construct</b>	In this study, the preferred term for a Data Model's primitive element.
<b>creativity</b>	Coming up with new and innovative ideas

<b>data</b>	Data correspond to discrete, recorded facts about phenomena from which we gain information about the world. However, they do not always correspond to concrete or actual facts. Sometimes they are imprecise or they describe things that have never happened (e.g. idea). For our purposed data correspond to descriptions of any phenomena or idea that a person considered worth formulating and recording. Data will be of interest to us if they are worth not only thinking about but also worth recording in a somewhat precise manner. (taken largely from Tsichritzis and Lochovsky, 1982).
<b>data model</b>	A representation of a particular set of data created according to the syntax of a specific Data Model (see below). A data model will usually minimally include a graphical representation of the data and textual details. It may also be called an 'application model' or a 'user model'.
<b>Data Model</b>	A set of conventions used to represent a simplified, formal and highly abstracted view of data. A Data Model will define a set of primitive elements (or constructs) and a set of rules for regulating how the elements can be combined to represent data objects (the 'way of modelling'). Most Data Models will also have guidelines on how they should be represented formally ('way of communicating'). Some definitions of Data Model also insist that rules for manipulating the primitive elements should also be defined, as in the Relational Model.
<b>data modelling</b>	The process by which a data model is created by the use of a data modelling approach.
<b>data modelling approach</b>	Usually used to describe a generic type rather than a specific Data Model, e.g. the E-R approach describes the general characteristics of the Data Model rather than those of a particular version.  In this study, it is used to convey the combination of the 'way of modelling', the 'way of communicating' and the 'way of working'.
<b>data modelling facility</b>	The combination of the 'way of modelling' and 'way of communicating' inherent in the use of a specific Data Model.
<b>data modelling formalism</b>	As data modelling facility
<b>data modelling method</b>	A set of guidelines or procedures setting out the steps to be followed in creating a data model with a specific data modelling approach.
<b>data modelling methodology</b>	A predefined set of steps, together with a collection of tools that can be used to build a data model. The term is often used interchangeably with data modelling method. Wherever possible, its use has been avoided in this study.

<b>data modelling scheme</b>	As data modelling facility.
<b>data modelling technique</b>	As data modelling facility.
<b>datalogical model</b>	A view of a data model, which emphasises the structural qualities of the data. The primary function of a datalogical model is to represent the designed structure of the data.
<b>descriptive</b>	Relating to, or based upon description or classification rather than explanation or prescription.
<b>design</b>	<p>The creation of an artefact, in accordance with appropriate functional or aesthetic criteria. Design is often considered to be the action of discovering solutions to problems.</p> <p>In this study it is specifically relates to the rendition of the identified user information requirement into a structural form, which can form the basis of an appropriate electronic database.</p>
<b>design methodology</b>	A collection of modelling methods that provides techniques to undertake design. Where such a methodology is tailored to data design, it will usually include techniques for analysis.
<b>design model</b>	A formal and precise specification of all identified requirements for a system under development. The structures represented in the model will have been constrained by the type of Data Model to which the final physical database will have to conform.
<b>entity</b>	<p>In general terms, something that has a real or distinct existence.</p> <p>In data modelling terms, an abstracted view of a collection of properties (attributes) that described something of interest to the system under consideration.</p>
<b>E-R Model</b>	A Data Model that uses three primitive elements, entities, attributes and relationships to represents data of interest to a system.
<b>E-R/Relational hybrid model</b>	An application data model that has a relational structure and is graphically represented with a form of E-R notation. The 'entities' in such a model are, in fact, relations and the 'relationships' are primary-foreign key links.
<b>implementation model,</b>	A type of Data Model which can be implemented by commercially available DBMSs, e.g. Relational, Object-Oriented, Network Models.
<b>infological model</b>	A view of a data model, which focuses on the information content of the model. The primary function of an infological model is to facilitate communication between technical and non-technical users.
<b>logical data design</b>	The stage of database design that is concerned with the design of the paradigmatic data structures. In some approaches, it includes

<b>design</b>	the analysis of the data. It is specifically separated from physical data design.
<b>logical model</b>	The end result of logical data design. The term, design model is preferred in this study.
<b>meta-data</b>	Data about data, e.g. descriptions of the properties of entities rather than the data values that they represent.
<b>meta-data architecture</b>	A framework for organising a number of different data models often of different types. Together they form a coherent representation of the data within a system or an organisation, at the conceptual, paradigmatic and physical levels.
<b>method</b>	Either the specification of behaviour or the software that implements that behaviour, that is stored in an object in an object-oriented development.
<b>NIAM</b>	Natural Language Information Analysis Method. A comprehensive and prescriptive method for the construction of relational database, which uses a formal subset of natural language to identify the information requirements.
<b>ORM</b>	Object-Role Model(ing). A Data Model which uses the primitive elements of objects and roles to represent data of interest to a system.
<b>paradigm model</b>	A level of model within the meta-data architecture that conforms to the rules of a specific Data Model, e.g. Relational, but which has not been tuned for implementation in a specific DBMS.
<b>physical model</b>	A representation of a data model that has been tailored for implementation in a particular DBMSs, e.g. DB2, and which includes the programming statements required to create all the database objects as well as the specification for those internal objects peculiar to the physical database, e.g. tablespaces and indexes.
<b>prescriptive</b>	Providing a well-defined pattern for usage. For example, a prescriptive method will detail the steps to be followed and the order in which they should be executed in order to achieve the desired end result.
<b>problem domain</b>	The area of an organisation that is under consideration as part of a database development activity.
<b>quality</b>	The notion of 'fitness for purpose'. Applied to database development is concerned both with 'building the system right' and with 'building the right system'.
<b>semantics</b>	The information content of the data model.
<b>tool</b>	An aid to an activity within the data modelling process, which is usually, but not essentially, an automated software aid.

<b>Universe of Discourse (UoD)</b>	The preferred term in this study for the ‘problem domain’.
<b>user requirement statement of -</b>	This is the users’ perception of the system that the users want and includes the set of functional and information requirements that the user will demand of a system.
<b>way of communicating</b>	A formal definition of the user requirement
<b>way of modelling</b>	This describes the form in which the models are to be communicated to human beings, usually in some form of graphical notation. Although models have the same way of modelling they can use different notation.
<b>way of working</b>	This describes the constructs, together with their properties and the permitted relationships between them. In other words it provides the grammar and syntax of the ‘language’ in which the models are to be expressed
<b>way of working</b>	This defines and orders the tasks and sub-tasks that are to be performed and also provides guidelines and heuristics on how these tasks should be carried out.



# Appendices



# Appendix 1

## InfoModeler™ transformations

On page 76 of this study it is suggested that InfoModeler™ will create an equivalent relational structure for unary and binary fact types which carry the same general meaning, thus allowing the analyst to retain the unary form if this is a more intuitive. The following examples illustrate this and are based on the sentence, 'Hedgehogs hibernate'

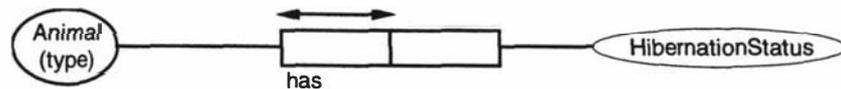
**Example 1** The animal type 'hedgehog' hibernates.



```

CREATE TABLE Animal (
  Type          varchar      (20) NOT NULL,
  /** Animal is ... **/
  Hibernates    logical      NOT NULL,
  /** Indicates whether Animal hibernates or not **/
  PRIMARY KEY (Type))
  
```

**Example 2** The animal type 'hedgehog' has HibernationStatus of 'H'



```

CREATE TABLE Animal (
  Type          varchar      (20) NOT NULL,
  /** Animal that has HibernationStatus. **/
  HibernationStatus logical    NULL,
  /** HibernationStatus that Animal has. **/
  PRIMARY KEY (Type))
  
```



## Appendix 2 - NaLER Definition

### Language

The NaLER sentences described in Chapter 10 are constructed with the elements described below. There are a number of primitive constructs some of which are *constants* and some *variables*. These primitives can be combined to create *sentences*, some of which are themselves constructed from subset combinations termed *phrases*.

#### Primitives

##### *Constants*

All constants are italicised. They are,

<i>Each</i>	A string which signifies that every instance of the entity participates in the 'fact', which follows it.
<i>Is uniquely identified by</i>	A string which describes the specific relationship between an entity and its primary key attribute.
<i>is a</i>	A string which describes the specific relationship between a specialised entity and the entity which is the generalisation of the first named entity.
<i>must have only one</i>	A string which denotes the relationship between an entity and an attribute which it contains, where a data value is always required in the attribute.
<i>may have only one</i>	A string which denotes the relationship between an entity and an attribute which it contains where the attribute may hold a null value.
<i>and</i>	A string which is used as a conjunction between two <i>phrases</i> .
<i>(</i>	A string which is used to signify the beginning of a <i>primary key variable</i> .

- ) A string which is used to signify the end of a *primary key variable*.
- .
- A string which identifies the completion of a sentence.

### **Variables**

All variables are delineated by arrow brackets. They are,

- <entity name>** The name of an entity on the E-R/R diagram
- <super entity name>** The name of a super-entity on an E-R/R diagram
- <sub entity name>** The name of a sub-entity on an E-R/R diagram
- <primary key>** The name of the primary key attribute(s) of the entity which immediately precedes it. If it is a composite primary key, the attribute names are separated by a comma.
- <attribute name>** The name of an attribute within the named entity
- <relationship name>** The name of a relationship which links the two named entities
- <optionality>** This describes whether or not the participation of an entity in a relationship is shown on the diagram as mandatory or not. It can have two values 'may' or 'must'
- <cardinality>** This describes the degree of the relationship as shown on the diagram.

**Phrases**

**EP** (Entity Phrase)            <entity name> (<primary key >)

**RP** (Relationship Phrase)    <optionality> <relationship-name> <cardinality>

**Sentences****1. Primary Key Sentence**

*Each <entity-name> is uniquely identified by <primary key>.*

**2. Attribute Sentence**

*Each EP must have only one <attribute name >.*

**3. Super-sub Sentence**

*Each <sub-entity-name>(<primary key>) is a <super-entity-name> (<primary key>).*

**4. Binary sentence**

*Each EP RP EP .*

**5. Ternary sentence ( Type 1)**

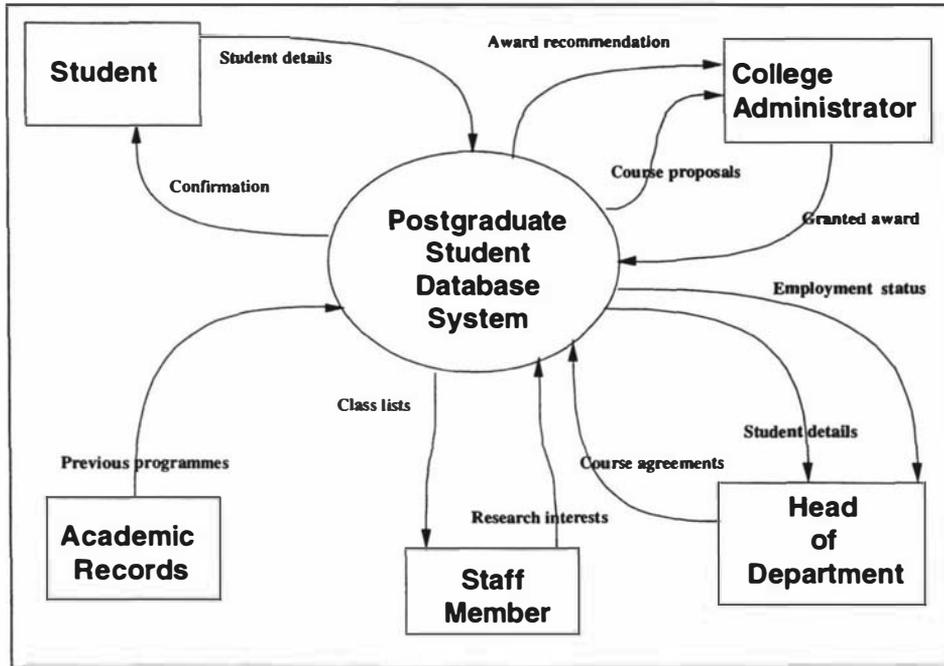
*Each EP RP EP and RP EP .*

**6. Ternary sentence (Type 2)**

*Each EP RP EP and RP EP and RP EP .*

\*

## Appendix 3 - ISPG Context Diagram



<b>Student details</b>	Includes all identification and contact details, current enrolment, future plans and research interests
<b>Confirmation</b>	Report of all information held for an individual student by the system
<b>Award recommendation</b>	Complete information on a student's results for a completed programme of study including the grade point average and a recommendation that the award be granted with a particular grade
<b>Course proposals</b>	Details of a proposed programme of study including all papers and the time periods in which they will be studied.
<b>Granted award</b>	Notification that an award has been given with a specific grade
<b>Employment status</b>	Notification of a students current employment status in the university
<b>Course agreements</b>	Record of any decisions made by the head of department pertaining to a students programme
<b>Research interests</b>	The areas of research interest of a member of staff in which research supervision will be accepted
<b>Class lists</b>	List of all students enrolled in a particular paper together with their contact details
<b>Previous programmes</b>	Record of previous programmes of study and awards gained together with grades where appropriate



# Appendix 4 - Analysis Documentation

## Infological

### 4.1 Initial Sentences

#### 4.1.1 Student View

1. A student enrolls in a paper in a particular semester of a particular year.
2. A paper has a specified number of points
3. A paper is run by a member of staff
4. A student enrolls in an endorsed programme in a particular year.
5. A programme requires a certain number of points
6. A student has a name.
7. A student has an address.
8. A student has a unique student id.
9. A student may have a unique email address.
10. A student may have a phone number.
11. A student requires approval for a programme of study.
12. A student enrolled in a Special Topic paper requires content approval.
13. A student enrolled in a Research Paper has a research supervisor.
14. A Special Topic is a type of paper.
15. A Research Paper is a type of paper.

#### **4.1.2 Paper Co-ordinator View**

1. Each paper is identified by a number
2. Each paper has a title
3. Each paper has a points value
4. One paper may be offered in any semester in any year.
5. A staff member may offer more than one paper at a time
6. Many students can be enrolled in such a paper
7. Each student has a unique student id.
8. Each student has a name
9. Each student has contact details
10. Each student has a final grade for such a paper
11. Each student is enrolled with a College of the university
12. If such a paper is a Special Topic then  
each student has an approved content
13. If such a paper is a Research Project then  
each student has a research title  
the student in the paper may have additional supervisors.

### 4.1.3 Head of Department View

1. A student is enrolled in a programme within a College for a particular time period
2. A student's enrolment in such a programme requires HoD approval
3. A student's enrolment in such a programme College approval.
4. A student is enrolled in papers in such a programme.
5. A student has a name
6. A student has one or more previous academic qualifications or
7. A student has been granted graduate status
8. A student enrolled in a research paper has a supervisor
9. A student enrolled in a special topic needs approval for the content
10. A student has a result for a paper
11. A student may have several areas of research interest
12. A student may be employed within the department in some capacity.
13. A staff member can be a co-ordinator for several papers at the same time.
14. A staff member may be a supervisor of several research projects at any one time
15. A staff member has research interests
16. A student's research paper has a project title.

#### **4.1.4 College Administrator View**

- 1. A student has a unique identifier**
- 2. A student has a full name**
- 3. A student has an address**
- 4. A student has a programme**
- 5. A student has an annual schedule of proposed papers for a programme**
- 6. A student has Head of Department of approval for programme**
- 7. A student has a total number of points for a programme**
- 8. A student has a schedule of completed papers for a programme for each year of that programme**
- 9. A student has a grade for each completed paper**
- 10. A paper within each programme has a points value**
- 11. A student has a grade point score for each completed paper**
- 12. A student has a grade point average for a completed programme**
- 13. A student has a supervisor for each completed research paper**
- 14. A student has a title for each completed research paper**
- 15. A student has a grade recommendation for an award**

## 4.2 Qualified Fact Types

### 4.2.1 Student View

#### Facts:

1. Paper is run by Staff\_Member  
Each Paper is run by at most one Staff\_Member
2. Paper is worth Points  
Each Paper is worth at most one Points
3. Programme requires Total\_Points  
Every Programme requires exactly one Total\_Points
4. Student enrolls in Paper in Semester in Year  
Each Student, Paper, Semester, Year combination is unique
5. Student enrolls in Programme with Endorsement in Year  
Each Student, Programme, Year combination occurs with at most one Endorsement
6. Student has Address  
Each Student has at most one Address
7. Student has Email\_Address  
Each Student has at most one Email\_Address and  
Each Email\_Address has at most one Student that has it
8. Student has Name  
Each Student has at most one Name
9. Student has Phone\_no  
Each Student has at most one Phone\_no
10. Student has Research\_Supervisor for Research\_Paper for Year  
Each Student, Research\_Paper, Year combination occurs with at most one Research\_Supervisor
11. Student requires Approval for Programme  
Each Student, Programme combination occurs with at most one Approval
12. Student requires Approved\_Content for Special\_Topic  
Each Student, Special\_Topic combination occurs with at most one Approved\_Content

#### 4.2.2.Paper Co-ordinator View

##### Facts:

1. Paper has Title  
Every Paper has exactly one Title
2. Paper offered in Semester and Year  
Each Paper, Semester, Year combination is unique
3. PaperSemesterYear has a Type  
Every PaperSemesterYear has a exactly one Type
4. PaperSemesterYear has Points\_value  
Every PaperSemesterYear has exactly one Points\_value
5. PaperSemesterYear run by Staff\_member  
Every PaperSemesterYear run by at least one Staff\_member and  
Each Staff\_member has zero or more PaperSemesterYear that run by it
6. Student enrolls in PaperSemesterYear  
Every Student enrolls in at least one PaperSemesterYear and  
Every PaperSemesterYear has at least one Student that enrolls in it
7. Student has a Name  
Every Student has a exactly one Name and  
Each Name has at most one Student that has a it
8. Student has Contact\_details  
Each Student has at most one Contact\_details
9. Student is enrolled with College  
Every Student is enrolled with exactly one College
10. StudentPaperSemesterYear awarded a Result  
Each StudentPaperSemesterYear awarded a at most one Result
11. StudentPaperSemesterYear has Approved\_Content  
Each StudentPaperSemesterYear has at most one Approved\_Content
12. StudentPaperSemesterYear has Project\_Title  
Each StudentPaperSemesterYear has at most one Project\_Title and  
Each Project\_Title has at most one StudentPaperSemesterYear that has it
13. StudentPaperSemesterYear has Supervisor  
Each StudentPaperSemesterYear has zero or more Supervisor and  
Each Supervisor has zero or more StudentPaperSemesterYear that has it
14. Supervisor has Research\_interest  
Each Supervisor has zero or more Research\_interest and  
Each Research\_interest has zero or more Supervisor that has it

### 4.2.3 Head of Department View

**Facts:**

1. Paper offered by Staff\_member in Semester in Year  
Each Paper, Staff\_member, Semester, Year combination is unique
2. Staff\_member has Research\_interest  
Each Staff\_member has zero or more Research\_interest and  
Each Research\_interest has zero or more Staff\_member that has it
3. Student employed as Position  
Each Student employed as at most one Position
4. Student enrolls in Programme  
Every Student enrolls in at least one Programme and  
Every Programme has at least one Student that enrolls in it
5. Student granted Graduate\_status  
Each Student granted at most one Graduate\_status
6. Student has gained Previous\_qualification  
Each Student has gained zero or more Previous\_qualification and  
Each Previous\_qualification has zero or more Student that has gained it
7. Student has Name  
Each Student has at most one Name
8. Student has Research\_interest  
Each Student has zero or more Research\_interest and  
Each Research\_interest has zero or more Student that has it
9. StudentEnrolment gains Result  
Each StudentEnrolment gains at most one Result
10. StudentEnrolment has Project\_title  
Each StudentEnrolment has at most one Project\_title and  
Each Project\_title has at most one StudentEnrolment that has it
11. StudentEnrolment has Supervisor  
Each StudentEnrolment has zero or more Supervisor and  
Each Supervisor has zero or more StudentEnrolment that has it
12. StudentEnrolment needs Approved\_content  
Each StudentEnrolment needs at most one Approved\_content
13. StudentProgramme consists of Paper  
Each StudentProgramme consists of zero or more Paper and  
Each Paper has zero or more StudentProgramme that consists of it

14. StudentProgramme is for Time\_Period  
Each StudentProgramme is for at most one Time\_Period
15. StudentProgramme is within College  
Each StudentProgramme is within at most one College
16. StudentProgramme requires College\_approval  
Each StudentProgramme requires at most one College\_approval
17. StudentProgramme requires Department\_Approval  
Each StudentProgramme requires at most one Department\_Approval
18. StudentProgrammePaper is for Semester in Year  
StudentProgrammePaper is mandatory  
Each StudentProgrammePaper, Semester, Year combination is unique

#### 4.2.4 College Administrator View

##### Facts:

1. CompletedPaper has Grade  
Every CompletedPaper has exactly one Grade
2. CompletedPaper has Grade\_point\_score \*  
Every CompletedPaper has exactly one Grade\_point\_score  
Derived by rule 'value of grade \* points value (eg If grade = A and points = 10 then score = 80'
3. CompletedPaper has Points  
Every CompletedPaper has exactly one Points
4. CompletedPaper has Project\_title  
Each CompletedPaper has at most one Project\_title and  
Each Project\_title has at most one CompletedPaper that has it
5. CompletedPaper has Supervisor  
Each CompletedPaper has at most one Supervisor
6. Student enrolls in Programme  
Each Student enrolls in zero or more Programme and  
Each Programme has zero or more Student that enrolls in it
7. Student has Address  
Each Student has at most one Address
8. Student has Full\_name  
Each Student has at most one Full\_name and  
Each Full\_name has at most one Student that has it
9. StudentProgramme completes Paper in Year  
Each StudentProgramme, Paper, Year combination is unique
10. StudentProgramme has Grade\_point\_average \*  
Each StudentProgramme has at most one Grade\_point\_average  
Derived by rule 'sum of grade point score/sum of points'
11. StudentProgramme has HoD\_approval  
Every StudentProgramme has exactly one HoD\_approval
12. StudentProgramme has Points\_total \*  
Each StudentProgramme has at most one Points\_total  
Derived by rule 'Sum of points for all required papers in programme'
13. StudentProgramme has Recommended\_grade  
Each StudentProgramme has at most one Recommended\_grade
14. StudentProgramme proposes Paper in Year  
Each StudentProgramme, Paper, Year combination is unique

### 4.3. Example sentences

#### 4.3.1 Student View

##### Facts:

##### 1. Paper is run by Staff\_Member

Each Paper is run by at most one Staff\_Member

##### Examples:

Paper '57.720' is run by Staff\_Member 'Larry Haist'

Paper '57.721' is run by Staff\_Member 'Jon Patrick'

Paper '57.794' is run by Staff\_Member 'Jon Patrick'

##### 2. Paper is worth Points

Each Paper is worth at most one Points

##### Examples:

Paper '57.720' is worth Points '15'

Paper '57.721' is worth Points '10'

Paper '57.722' is worth Points '10'

##### 3. Programme requires Total\_Points

Every Programme requires exactly one Total\_Points

##### Examples:

Programme 'MBS' requires Total\_Points '200'

Programme 'DiplnfSci' requires Total\_Points '90'

Programme 'MSc' requires Total\_Points '200'

##### 4. Student enrolls in Paper in Semester in Year

Each Student, Paper, Semester, Year combination is unique

##### Examples:

Student '98010101' enrolls in Paper '57.794' in Semester '1' in Year '1998'

Student '98010101' enrolls in Paper '57.794' in Semester '2' in Year '1998'

Student '98010101' enrolls in Paper '57.794' in Semester '1' in Year '1999'

Student '98020202' enrolls in Paper '57.794' in Semester '1' in Year '1998'

##### 5. Student enrolls in Programme with Endorsement in Year

Each Student, Programme, Year combination occurs with at most one Endorsement

##### Examples:

Student '98010101' enrolls in Programme 'BSc(Hons)' with Endorsement 'Information Systems' in Year '1998'

Student '98020202' enrolls in Programme 'BSc(Hons)' with Endorsement 'Information Systems' in Year '1998'

Student '98010101' enrolls in Programme 'PhD' with Endorsement 'Information Systems' in Year '1998'

Student '98010101' enrolls in Programme 'PhD' with Endorsement 'Information Systems' in Year '1999'

Student '98030303' enrolls in Programme 'DiplnfSci' with Endorsement 'Information Systems' in Year '1999'

Student '98030303' enrolls in Programme 'DiplnfSci' with Endorsement 'Computing' in Year '1997'

##### 6. Student has Address

Each Student has at most one Address

##### Examples:

Student '98010101' has Address '59 Main St, Palmerston North'

Student '98020202' has Address '10 Cobden St, Feilding'

Student '96332211' has Address '59 Main St Palmerston North'

7. Student has Email\_Address

Each Student has at most one Email\_Address and  
Each Email\_Address has at most one Student that has it

Examples:

Student '98010101' has Email\_Address 'C.Atkins@massey.ac.nz'  
Student '98020202' has Email\_Address 'L.Weston@massey.ac.nz'

8. Student has Name

Each Student has at most one Name

Examples:

Student '98010101' has Name 'Clare Atkins'  
Student '98020202' has Name 'Liz Weston'

9. Student has Phone\_no

Each Student has at most one Phone\_no

Examples:

Student '98010101' has Phone\_no '350-4206'  
Student '98020202' has Phone\_no '350-5217'

10. Student has Research\_Supervisor for Research\_Paper for Year

Each Student, Research\_Paper, Year combination occurs with at most one Research\_Supervisor

Examples:

Student '98010101' has Research\_Supervisor 'Jon Patrick' for Research\_Paper '57.799' for Year '199'  
Student '98010101' has Research\_Supervisor 'Roger Tagg' for Research\_Paper '57.800' for Year '199'  
Student '98010101' has Research\_Supervisor 'Jon Patrick' for Research\_Paper '57.900' for Year '199'  
Student '98020202' has Research\_Supervisor 'Jon Patrick' for Research\_Paper '57.800' for Year '199'

11. Student requires Approval for Programme

Each Student, Programme combination occurs with at most one Approval

Examples:

Student '98010101' requires Approval 'Y' for Programme 'BSc(Hons)'  
Student '98010101' requires Approval 'N' for Programme 'DiplInfSci'  
Student '98010101' requires Approval 'Y' for Programme 'PhD'

12. Student requires Approved\_Content for Special\_Topic

Each Student, Special\_Topic combination occurs with at most one Approved\_Content

Examples:

Student '98010101' requires Approved\_Content '331 + essay' for Special\_Topic '57.794'  
Student '98020202' requires Approved\_Content '331 + essay' for Special\_Topic '57.794'  
Student '98010101' requires Approved\_Content '332 + 341' for Special\_Topic '57.795'

### 4.3.2 Paper Co-ordinator View

#### Facts:

#### 1. Paper has Title

- Every Paper has at least one Title
- Each Paper has at most one Title

#### Examples:

- Paper '57.720' has Title 'Information Systems Research Methods'
- Paper '57.722' has Title 'Semantic Modelling'
- Paper '57.794' has Title 'Special Topic in IS'
- Paper '57.795' has Title 'Special Topic in IS'

#### 2. Paper offered in Semester and Year

- Each Paper, Semester, Year combination is unique

#### Examples:

- Paper '57.720' offered in Semester '1' and Year '1998'
- Paper '57.720' offered in Semester '1' and Year '1999'
- Paper '57.794' offered in Semester '1' and Year '1998'
- Paper '57.794' offered in Semester '2' and Year '1998'
- Paper '57.799' offered in Semester '2' and Year '1998'
- Paper '57.721' offered in Semester '1' and Year '1998'

#### 3. PaperSemesterYear has a Type

- Every PaperSemesterYear has a at least one Type
- Each PaperSemesterYear has a at most one Type

#### Examples:

- PaperSemesterYear '57.794,1,1998' has a Type 'Special Topic'
- PaperSemesterYear '57.794,2,1998' has a Type 'Special Topic'
- PaperSemesterYear '57.799,2,1998' has a Type 'Research'

#### 4. PaperSemesterYear has Points\_value

- Every PaperSemesterYear has at least one Points\_value
- Each PaperSemesterYear has at most one Points\_value

#### 5. PaperSemesterYear run by Staff\_member

- Every PaperSemesterYear run by at least one Staff\_member
- Each PaperSemesterYear, Staff\_member combination is unique

#### Examples:

- PaperSemesterYear '57.720,1,1998' run by Staff\_member 'LH'
- PaperSemesterYear '57.799,2,1998' run by Staff\_member 'LH'
- PaperSemesterYear '57.799,2,1998' run by Staff\_member 'CA'
- PaperSemesterYear '57.721,1,1998' run by Staff\_member 'LH'

#### 6. Student enrolls in PaperSemesterYear

- Every Student enrolls in at least one PaperSemesterYear
- Every PaperSemesterYear has at least one Student that enrolls in it
- Each Student, PaperSemesterYear combination is unique

#### Examples:

- Student '97010101' enrolls in PaperSemesterYear '57.720,1,1997'
- Student '97020202' enrolls in PaperSemesterYear '57.720,1,1997'
- Student '97010101' enrolls in PaperSemesterYear '57.721,1,1997'
- Student '1999' enrolls in PaperSemesterYear '98010101,57.800,12'
- Student '1999' enrolls in PaperSemesterYear '98010101,57.800,12'
- Student '1998' enrolls in PaperSemesterYear '98020202,57.799,1'

7. Student has a Name

Every Student has a at least one Name

Each Student has a at most one Name

Each Name has at most one Student that has a it

Examples:

Student '98010101' has a Name 'Mike Ryder'

Student '98020202' has a Name 'Liz Weston'

8. Student has Contact\_details

Each Student has at most one Contact\_details

Examples:

Student '98010101' has Contact\_details '59,Main St 355-5555'

Student '98020202' has Contact\_details '10 Duna Place 354-6677'

Student '96020202' has Contact\_details '59,Main St 355-5555'

9. Student is enrolled with College

Every Student is enrolled with at least one College

Each Student is enrolled with at most one College

Examples:

Student '98010101' is enrolled with College 'Science'

Student '98020202' is enrolled with College 'Business'

Student '97010101' is enrolled with College 'Science'

10. StudentPaperSemesterYear awarded a Result

Each StudentPaperSemesterYear awarded a at most one Result

Examples:

StudentPaperSemesterYear '98010101,57.720,1,1998' awarded a Result 'A'

StudentPaperSemesterYear '98010101,57.721,1,1998' awarded a Result 'A'

StudentPaperSemesterYear '98020202,57.720,1,1998' awarded a Result 'B'

11. StudentPaperSemesterYear has Approved\_Content

Each StudentPaperSemesterYear has at most one Approved\_Content

Examples:

StudentPaperSemesterYear '98010101,57.794,1,1998' has Approved\_Content '331+essay'

StudentPaperSemesterYear '98010101,57.794,2,1998' has Approved\_Content 'work project'

StudentPaperSemesterYear '98020202,57.794,2,1998' has Approved\_Content 'work project'

12. StudentPaperSemesterYear has Project\_Title

Each StudentPaperSemesterYear has at most one Project\_Title

Each Project\_Title has at most one StudentPaperSemesterYear that has it

Examples:

StudentPaperSemesterYear '98010101,57.799,2,1998' has Project\_Title 'A case for CASE'

StudentPaperSemesterYear '98020202,57.799,2,1998' has Project\_Title 'An investigation of....'

13. StudentPaperSemesterYear has Supervisor

Each StudentPaperSemesterYear, Supervisor combination is unique

Examples:

StudentPaperSemesterYear '98010101,57.800,12,1999' has Supervisor 'RW'

StudentPaperSemesterYear '98010101,57.800,12,1999' has Supervisor 'CF'

StudentPaperSemesterYear '98020202,57.799,1,1998' has Supervisor 'RW'

14. Supervisor has Research\_interest

Each Supervisor, Research\_interest combination is unique

Examples:

Supervisor 'CF' has Research\_interest 'conceptual modelling'

Supervisor 'CA' has Research\_interest 'conceptual modelling'

Supervisor 'CA' has Research\_interest 'CASE tools'

### 4.3.3 Head of Department View

#### Facts:

#### 1. Paper offered by Staff\_member in Semester in Year

Each Paper, Staff\_member, Semester, Year combination is unique

#### Examples:

Paper '57.720' offered by Staff\_member 'LH' in Semester '1' in Year '1998'  
 Paper '57.799' offered by Staff\_member 'LH' in Semester '1' in Year '1998'  
 Paper '57.799' offered by Staff\_member 'LH' in Semester '2' in Year '1998'  
 Paper '57.799' offered by Staff\_member 'CA' in Semester '2' in Year '1998'  
 Paper '57.720' offered by Staff\_member 'LH' in Semester '1' in Year '1999'

#### 2. Staff\_member has Research\_interest

Each Staff\_member, Research\_interest combination is unique

#### Examples:

Staff\_member 'CF' has Research\_interest 'conceptual modelling'  
 Staff\_member 'CF' has Research\_interest 'Semantic modelling'  
 Staff\_member 'CA' has Research\_interest 'conceptual modelling'

#### 3. Student employed as Position

Each Student employed as at most one Position

#### Examples:

Student '98010101' employed as Position 'Graduate Assistant - half time'  
 Student '98020202' employed as Position 'Casual Assistant'  
 Student '98030303' employed as Position 'Graduate Assistant - half time'

#### 4. Student enrolls in Programme

Every Student enrolls in at least one Programme

Every Programme has at least one Student that enrolls in it

Each Student, Programme combination is unique

#### Examples:

Student '98010101' enrolls in Programme 'MBS'  
 Student '98010101' enrolls in Programme 'PhD'  
 Student '98020202' enrolls in Programme 'MBS'  
 Student '98020202' enrolls in Programme 'PhD'  
 Student '98020202' enrolls in Programme 'MSc'  
 Student '98040404' enrolls in Programme 'DiplnfSci'

#### 5. Student granted Graduate\_status

Each Student granted at most one Graduate\_status

#### Examples:

Student '98040404' granted Graduate\_status 'Y'  
 Student '98050505' granted Graduate\_status 'Y'

#### 6. Student has gained Previous\_qualification

Each Student, Previous\_qualification combination is unique

#### Examples:

Student '98010101' has gained Previous\_qualification 'BA(Hons)'  
 Student '98020202' has gained Previous\_qualification 'DipSci'  
 Student '98030303' has gained Previous\_qualification 'DipSci'  
 Student '98030303' has gained Previous\_qualification 'MSc'

## 7. Student has Name

Each Student has at most one Name

Examples:

Student '98010101' has Name 'Clare Atkins'

Student '98020202' has Name 'Mike Ryder'

## 8. Student has Research\_interest

Each Student, Research\_interest combination is unique

Examples:

Student '98040404' has Research\_interest 'conceptual modelling'

Student '98040404' has Research\_interest 'methodologies'

Student '98010101' has Research\_interest 'conceptual modelling'

## 9. StudentEnrolment gains Result

Each StudentEnrolment gains at most one Result

Examples:

StudentEnrolment '98040404,DiplInfSci,57.721,1,1998' gains Result 'B'

StudentEnrolment '98040404,DiplInfSci,57.794,1,1998' gains Result 'A'

StudentEnrolment '98040404,DiplInfSci,57.794,2,1998' gains Result 'A'

## 10. StudentEnrolment has Project\_title

Each StudentEnrolment has at most one Project\_title

Each Project\_title has at most one StudentEnrolment that has it

Examples:

StudentEnrolment '98010101,MBS,57.800,12,1997' has Project\_title 'A case for CASE'

StudentEnrolment '98010101,PhD,57.900,12,1999' has Project\_title 'An investigation into....'

## 11. StudentEnrolment has Supervisor

Each StudentEnrolment, Supervisor combination is unique

## 12. StudentEnrolment needs Approved\_content

Each StudentEnrolment needs at most one Approved\_content

Examples:

StudentEnrolment '98040404,DiplInfSci,57.794,1,1998' needs Approved\_content 'work projec

StudentEnrolment '98050505,DiplInfSci,57.794,1,1998' needs Approved\_content 'work projec

StudentEnrolment '98040404,DiplInfSci,57.794,2,1998' needs Approved\_content '332+ essay

## 13. StudentProgramme consists of Paper

Each StudentProgramme, Paper combination is unique

Examples:

StudentProgramme '98040404,DiplInfSci' consists of Paper '57.720'

StudentProgramme '98040404,DiplInfSci' consists of Paper '57.721'

StudentProgramme '98050505,DiplInfSci' consists of Paper '57.720'

StudentProgramme '98040404,DiplInfSci' consists of Paper '57.794'

StudentProgramme '98050505,DiplInfSci' consists of Paper '57.794'

StudentProgramme '98010101,MBS' consists of Paper '57.800'

## 14. StudentProgramme is for Time\_Period

Each StudentProgramme is for at most one Time\_Period

Examples:

StudentProgramme '98010101,MBS' is for Time\_Period '1997-1998'

StudentProgramme '98010101,PhD' is for Time\_Period '1998-2001'

StudentProgramme '98020202,PhD' is for Time\_Period '1998-2001'

## 15. StudentProgramme is within College

Each StudentProgramme is within at most one College

## Examples:

StudentProgramme '98010101,MBS' is within College 'Business'

StudentProgramme '98010101,PhD' is within College 'Business'

StudentProgramme '98020202,PhD' is within College 'Science'

## 16. StudentProgramme requires College\_approval

Each StudentProgramme requires at most one College\_approval

## Examples:

StudentProgramme '98010101,MBS' requires College\_approval 'Y'

StudentProgramme '98020202,MSc' requires College\_approval 'Y'

## 17. StudentProgramme requires Department\_Approval

Each StudentProgramme requires at most one Department\_Approval

## Examples:

StudentProgramme '98010101,PhD' requires Department\_Approval 'Y'

StudentProgramme '98020202,PhD' requires Department\_Approval 'Y'

## 18. StudentProgrammePaper is for Semester in Year

Every StudentProgrammePaper must participate in this relationship

Each StudentProgrammePaper, Semester, Year combination is unique

## Examples:

StudentProgrammePaper '98040404,DipInfSci,57.794' is for Semester '1' in Year '1998'

StudentProgrammePaper '98040404,DipInfSci,57.794' is for Semester '2' in Year '1998'

StudentProgrammePaper '98040404,DipInfSci,57.794' is for Semester '1' in Year '1999'

StudentProgrammePaper '98050505,DipInfSci,57.794' is for Semester '1' in Year '1998'

StudentProgrammePaper '98010101,MBS,57.800' is for Semester '12' in Year '1997'

StudentProgrammePaper '98010101,PhD,57.900' is for Semester '12' in Year '1999'

### 4.3.4 College Administrator View

#### Facts:

#### 1. CompletedPaper has Grade

Every CompletedPaper has exactly one Grade

#### Examples:

CompletedPaper '98010101,Diploma of Science,57.794,1998' has Grade 'B'

CompletedPaper '98010101,Diploma of Science,57.794,1999' has Grade 'A'

CompletedPaper '98020202,Diploma of Science,57.794,1998' has Grade 'A'

#### 2. CompletedPaper has Grade\_point\_score \*

Every CompletedPaper has exactly one Grade\_point\_score

Derived by rule 'value of grade \* points value (eg If grade = A and points = 10 then score = 80'

#### Examples:

CompletedPaper '98010101,Diploma of Science,57.794,1998' has Grade\_point\_score '80'

CompletedPaper '98010101,Diploma of Science,57.794,1999' has Grade\_point\_score '60'

CompletedPaper '98020202,Diploma of science,57.794,1998' has Grade\_point\_score '80'

#### 3. CompletedPaper has Points

Every CompletedPaper has exactly one Points

#### Examples:

CompletedPaper '98010101,Diploma of Science,57.720,1998' has Points '15'

CompletedPaper '98010101,Diploma of Science,57.721,1998' has Points '10'

CompletedPaper '98020202,Diploma of Science,57.720,1998' has Points '15'

#### 4. CompletedPaper has Project\_title

Each CompletedPaper has at most one Project\_title and

Each Project\_title has at most one CompletedPaper that has it

#### Examples:

CompletedPaper '98010101,Master of Science,57.799,1998' has Project\_title 'A case of CASE'

CompletedPaper '98020202,Master of Science,57.799,1998' has Project\_title 'An investigation of....'

#### 5. CompletedPaper has Supervisor

Each CompletedPaper has at most one Supervisor

#### Examples:

CompletedPaper '98010101,Master of Science,57.799,1998' has Supervisor 'Larry Haist'

CompletedPaper '98010101,Master of Science,57.800,1999' has Supervisor 'Ellen Rose'

CompletedPaper '98020202,Master of Science,57.799,1998' has Supervisor 'Larry Haist'

#### 6. Student enrolls in Programme

Each Student enrolls in zero or more Programme and

Each Programme has zero or more Student that enrolls in it

#### Examples:

Student '98010101' enrolls in Programme 'Master of Science'

Student '98010101' enrolls in Programme 'Doctor of Philosophy'

Student '98020202' enrolls in Programme 'Master of Science'

Student '98010101' enrolls in Programme 'Diploma of Social Sciences'

Student '98020202' enrolls in Programme 'Master of Science'

Student '98060606' enrolls in Programme 'Master of Science'

#### 7. Student has Address

Each Student has at most one Address

#### Examples:

Student '98010101' has Address '59 Main Street, Palmerston North'

Student '98020202' has Address '10 Duna Place, Feilding'

Student '98030303' has Address '59 Main Street, Palmerston North'

## 8. Student has Full\_name

Each Student has at most one Full\_name and

Each Full\_name has at most one Student that has it

Examples:

Student '98010101' has Full\_name 'Clare Frances Atkins'

Student '98020202' has Full\_name 'Michael Robert Ryder'

## 9. StudentProgramme completes Paper in Year

Each StudentProgramme, Paper, Year combination is unique

Examples:

StudentProgramme '98010101,Diploma of Science' completes Paper '57.794' in Year '1998'

StudentProgramme '98010101,Diploma of Science' completes Paper '57.721' in Year '1998'

StudentProgramme '98010101,Diploma of Science' completes Paper '57.794' in Year '1999'

StudentProgramme '98020202,Diploma of Science' completes Paper '57.794' in Year '1998'

StudentProgramme '98020202,Diploma of science' completes Paper '57.794' in Year '1998'

StudentProgramme '98010101,Diploma of Science' completes Paper '57.720' in Year '1998'

## 10. StudentProgramme has Grade\_point\_average \*

Each StudentProgramme has at most one Grade\_point\_average

Derived by rule 'sum of grade point score/sum of points'

Examples:

StudentProgramme '98010101,Diploma of Science' has Grade\_point\_average '2'

StudentProgramme '98010101,Master of Science' has Grade\_point\_average '4.5'

StudentProgramme '98020202,Diploma of Science' has Grade\_point\_average '2'

## 11. StudentProgramme has HoD\_approval

Every StudentProgramme has exactly one HoD\_approval

Examples:

StudentProgramme '98010101,Master of Science' has HoD\_approval 'IS HoD'

StudentProgramme '98010101, Diploma of Social Sciences' has HoD\_approval 'IS HoD'

StudentProgramme '98020202, Master of Science' has HoD\_approval 'IS HoD'

StudentProgramme '98060606,Master of Science' has HoD\_approval 'CS HoD'

## 12. StudentProgramme has Points\_total \*

Each StudentProgramme has at most one Points\_total

Derived by rule 'Sum of points for all required papers in programme'

Examples:

StudentProgramme '98010101,Diploma of Science' has Points\_total '95'

StudentProgramme '98010101,Master of Science' has Points\_total '200'

StudentProgramme '98020202,Diploma of Science' has Points\_total '90'

StudentProgramme '98020202,Master of Science' has Points\_total '200'

## 13. StudentProgramme has Recommended\_grade

Each StudentProgramme has at most one Recommended\_grade

Examples:

StudentProgramme '98010101,Diploma of Science' has Recommended\_grade 'Distinction'

StudentProgramme '98010101,Master of Science' has Recommended\_grade 'First Class Honours'

StudentProgramme '98020202,Master of Science' has Recommended\_grade 'Distinction'

## 14. StudentProgramme proposes Paper in Year

Each StudentProgramme, Paper, Year combination is unique

Examples:

StudentProgramme '98010101,Diploma of Science' proposes Paper '57.794' in Year '1998'

StudentProgramme '98010101,Diploma of Science' proposes Paper '57.794' in Year '1999'

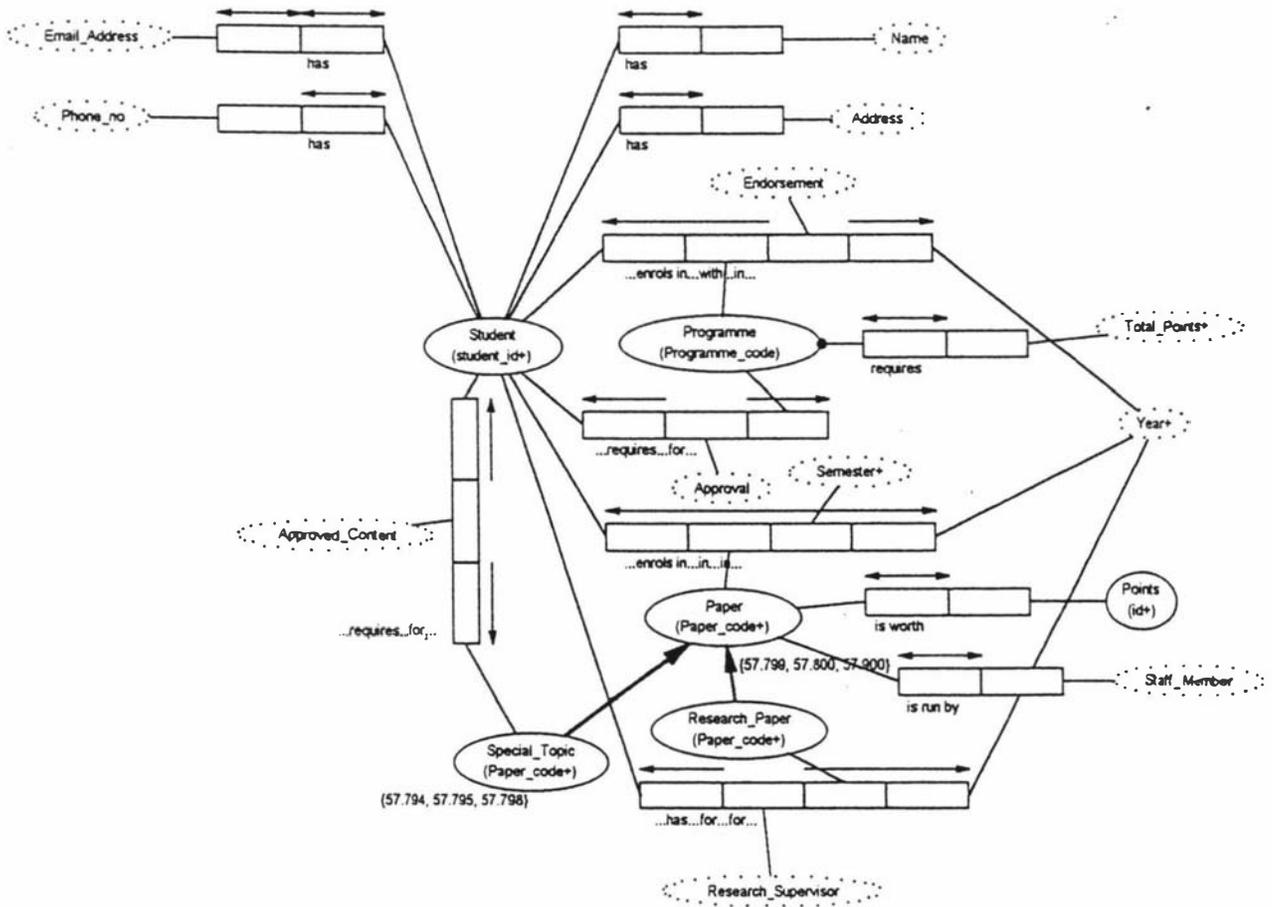
StudentProgramme '98020202,Diploma of Science' proposes Paper '57.794' in Year '1998'

StudentProgramme '98010101,Master of Science' proposes Paper '57.800' in Year '1999'

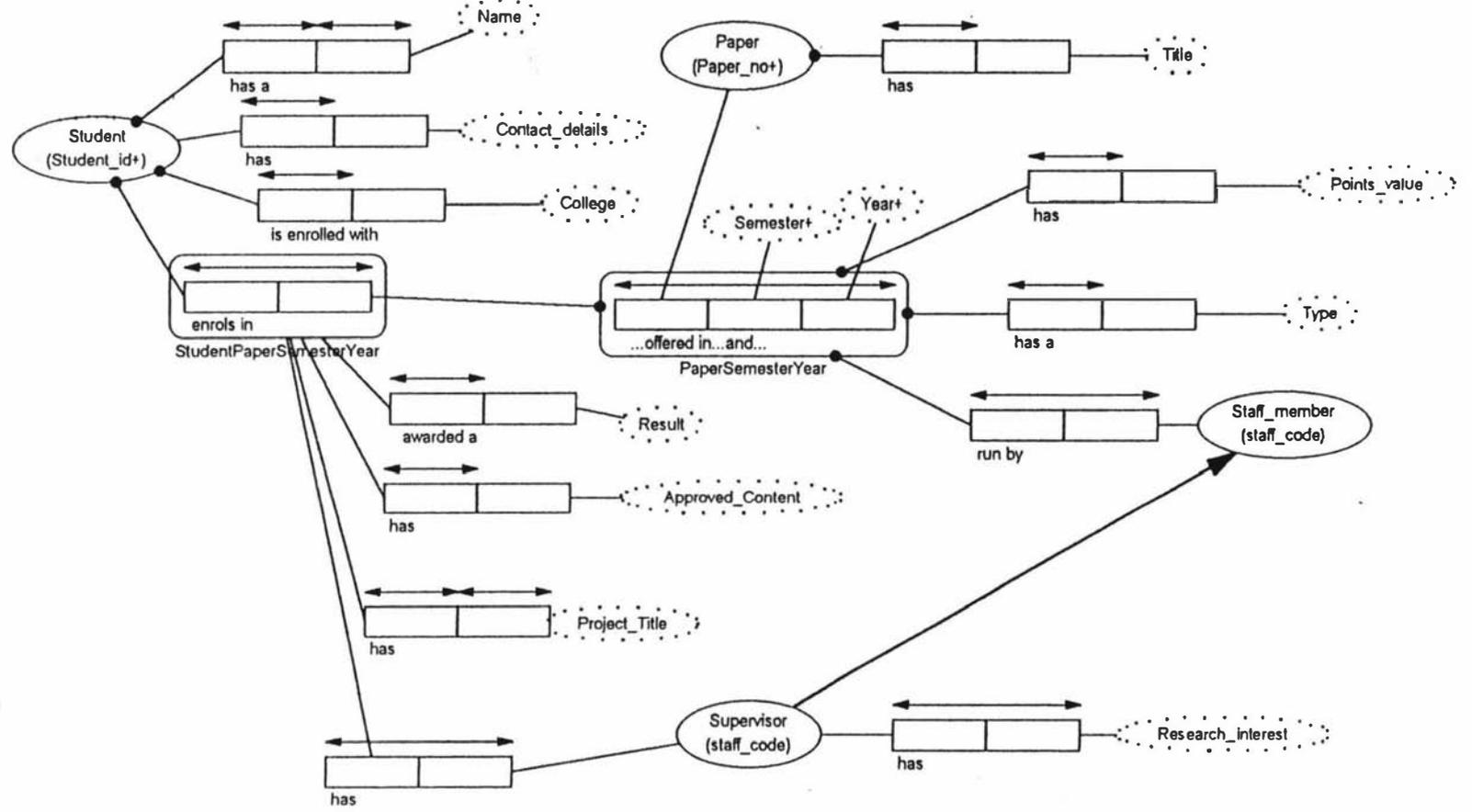
# Datological Model

## 4.4 ORM Diagrams

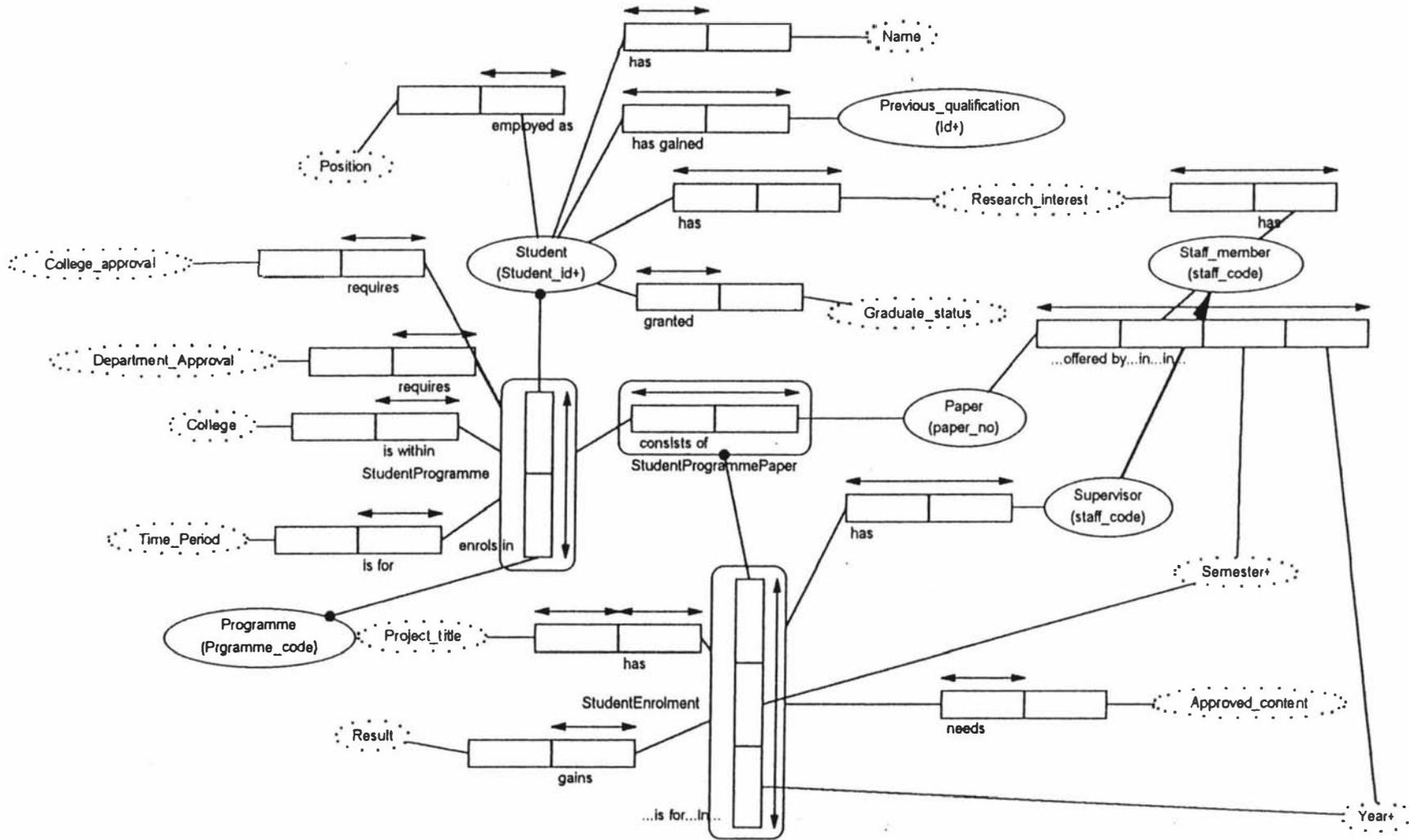
### 4.4.1 Student View



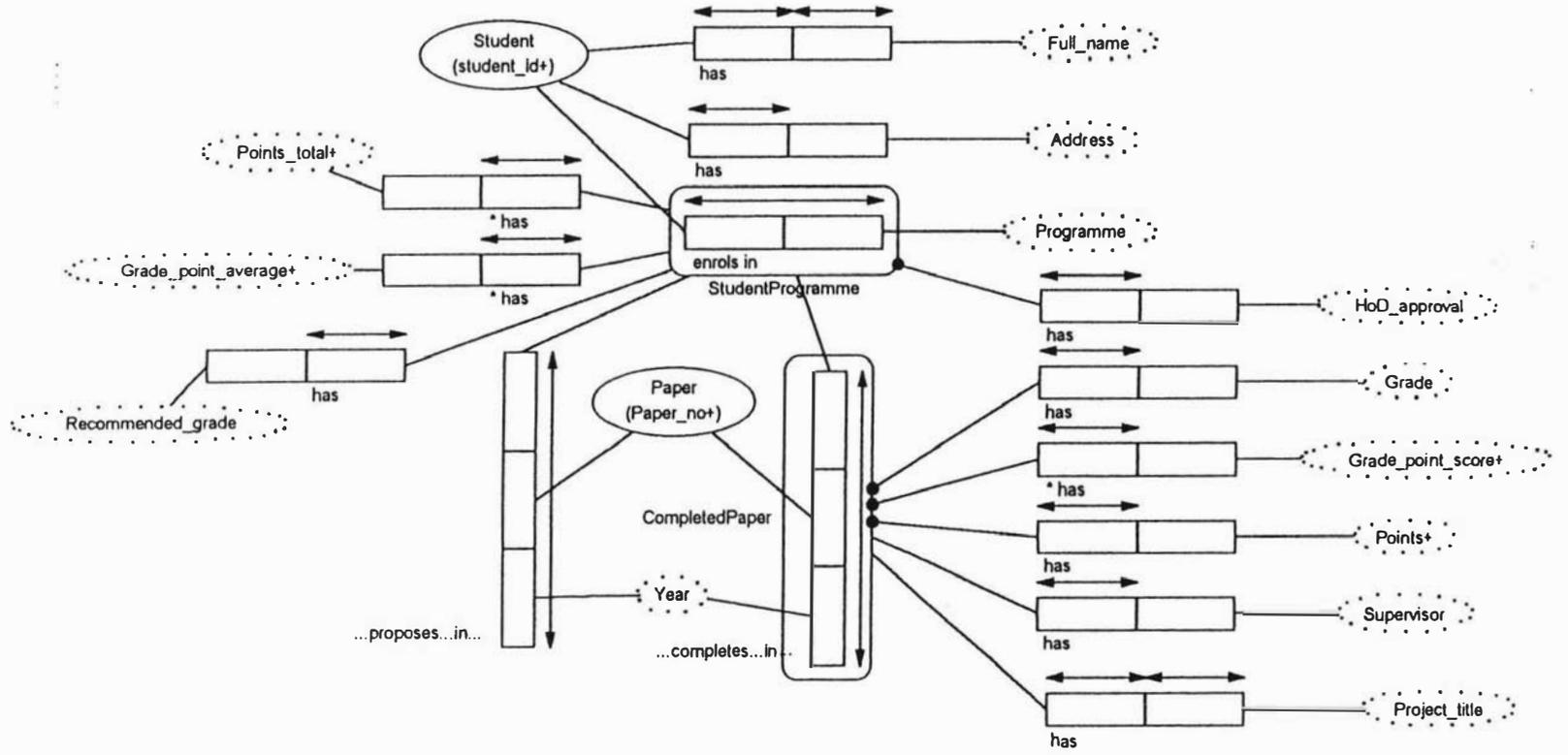
## 4.4.2 Paper Co-ordinator View



### 4.4.3 Head of Department View

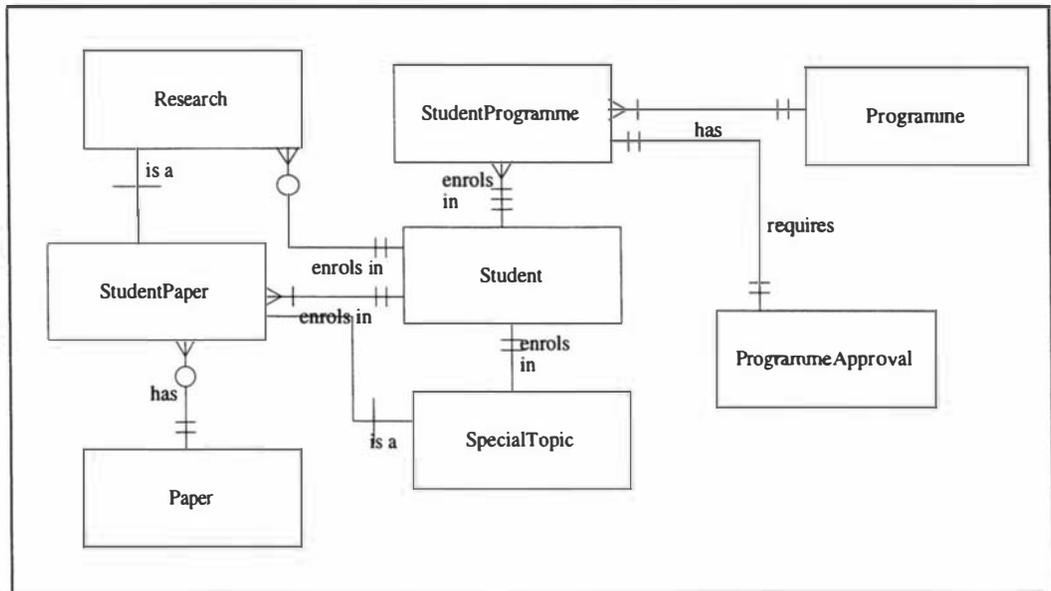


4.4.4 College Administrator View



# Appendix 5 - Initial Design

## 5.1 Initial design model - Student View



**ProgrammeApproval**

\*Student\_id  
 \*Programme\_code  
 Approval

**Research**

\*Student\_id  
 \*Paper\_code  
 \*Year  
 Staff\_code

**SpecialTopic**

\*Student\_id  
 \*Paper\_code  
 Approved\_content

**Student**

\*Student\_id  
 Name  
 Address  
 Email  
 Phone\_no

**StudentPaper**

\*Student\_id  
 \*Paper\_code  
 \*Year  
 \*Semester

**StudentProgramme**

\*Student\_id  
 \*Programme\_code  
 \*Year  
 Endorsement

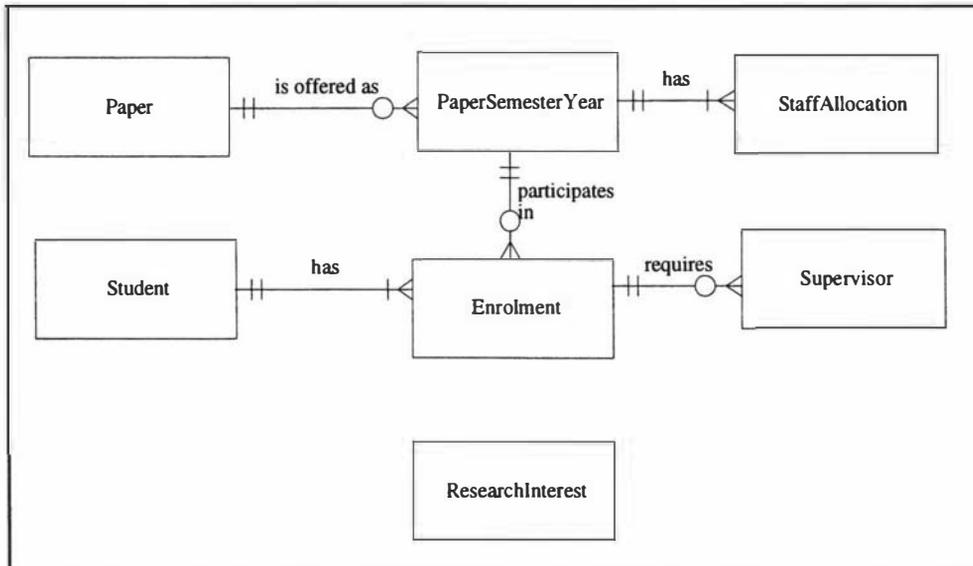
**Paper**

\*Paper\_code  
 Points  
 Staff\_member

**Programme**

\*Programme\_code  
 Total\_points

## 5.2 Initial design model - Paper Co-ordinator View



### Paper

\*Paper\_no  
Title

### PaperSemesterYear

\*Paper\_no  
\*Semester  
\*Year  
Points\_value  
Type

### ResearchInterest

\*Staff\_name  
\*Research\_interest

### Student

\*Student\_id  
Name  
Contact\_details  
Enrolled\_college

### Enrolment

\*Student\_id  
\*Paper\_no  
\*Semester  
\*Year  
Approved\_content  
Result  
Project\_title

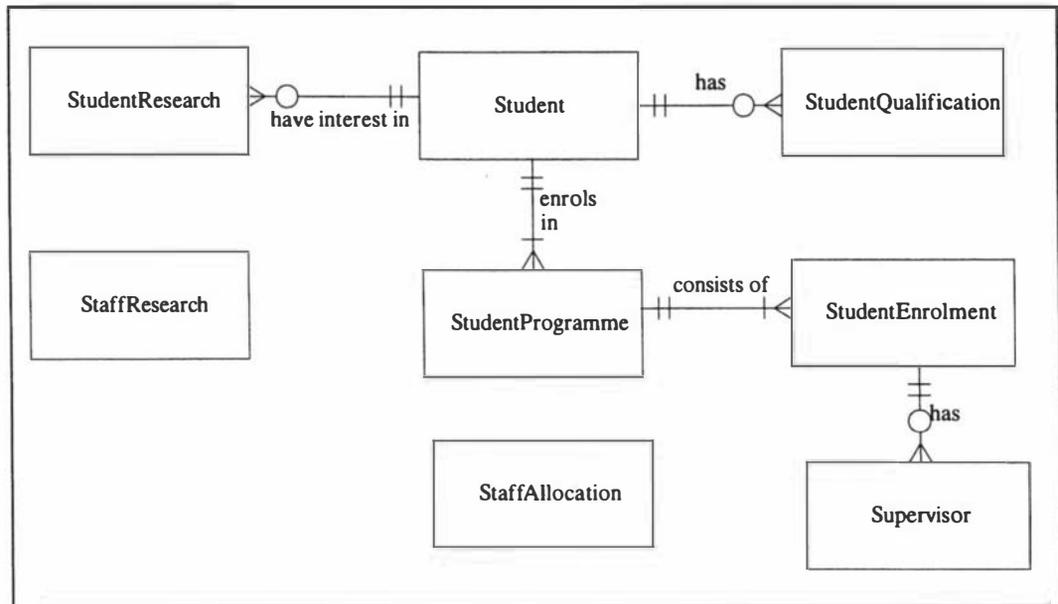
### StaffAllocation

\*Paper\_no  
\*Semester  
\*Year  
\*Staff\_name

### Supervisor

\*Student\_id  
\*Paper\_no  
\*Semester  
\*Year  
\*Staff\_name

### 5.3 Initial design model - Head of Department View



**StudentResearch**  
 \*Student\_id  
 \*Research\_interest

**StaffAllocation**  
 \*Paper\_no  
 \*Semester  
 \*Year  
 \*Staff\_name

**StudentQualification**  
 \*Student\_id  
 \*Qualification

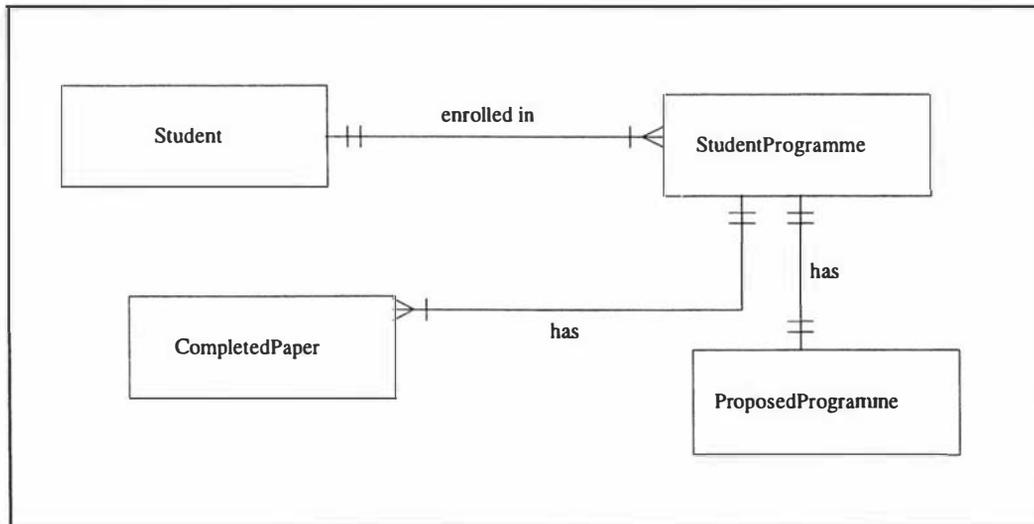
**Student**  
 \*Student\_id  
 Name  
 Graduate\_status  
 Position

**StudentEnrolment**  
 \*Student\_id  
 \*Programme\_code  
 \*Paper\_code  
 \*Semester  
 \*Year  
 Approved\_content  
 Result  
 Project\_title

**StudentProgramme**  
 \*Student\_id  
 \*Programme\_code  
 \*Paper\_no  
 \*Semester  
 \*Year  
 \*Staff\_name

**ResearchInterest**  
 \*Staff\_name  
 \*Research\_interest

### 5.4 College Administrator View



#### Student

\*Student\_id  
Full\_name  
Address

#### ProposedProgramme

\*Student\_id  
\*Programme  
\*Paper\_no  
\*Year

#### StudentProgramme

\*Student\_id  
\*Programme  
HoD\_approval  
Recommended\_grade

#### CompletedPaper

\*Student\_id  
\*Programme  
\*Paper\_no  
\*Year  
Grade  
Points  
Supervisor  
Project\_title

## 5.5 Initial Global Design Model

### Entity/Attribute List

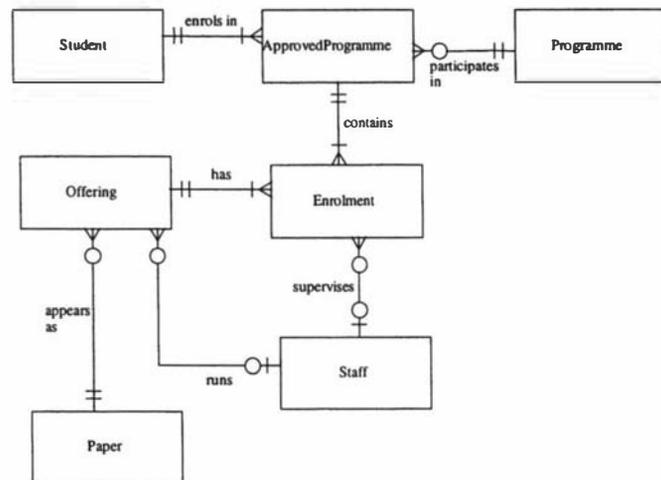
As there were a number of redundant and overlapping relationships implicit in this draft schema, it was not possible to create a useful diagram at this point.

Entity Name	Primary Key	Other Attributes
Paper	Paper_code	Title, Points
PaperSemesterYear	Paper_code, Semester, Year	Points_value, Type
Programme	Programme_code	Points_value
StaffAllocation	Paper_code, Semester, Year, Staff_name	
StaffResearch	Staff_name, Research_interest	
Student	Student_id	Full_name, Address, Email_address, Phone_no, Enrolled_college, Graduate_status, Position
Supervisor	Student_id, Paper_code, Semester, Year Staff_name	
ProposedProgramme	Student_id, Programme, Paper, Year	
CompletedPaper	Student_id, Programme, Paper, Year	Grade, Points, Supervisor, Project_title
ProgrammeApproval	Student_id, Programme_code	HoD_approval, Recommended_grade
Enrolment	Student_id, Programme_code, Paper, Semester, Year	Approved_content, Result, Project_title
StudentProgramme	Student_id, Programme_code, Paper, Semester, Year, Staff_name	
StudentQualification	Student_id, Qualification	
StudentResearch	Student_id, Research_interest	



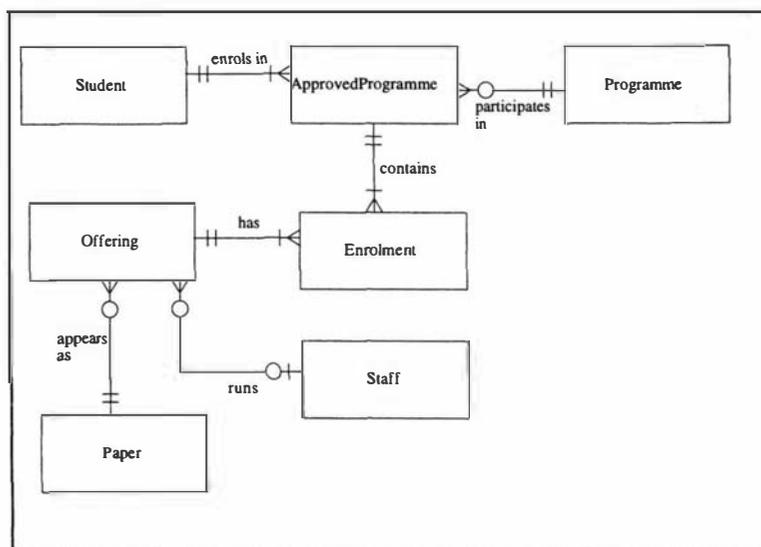
# Appendix 6 - Design in Progress

## 6.1 Second Draft Design Model



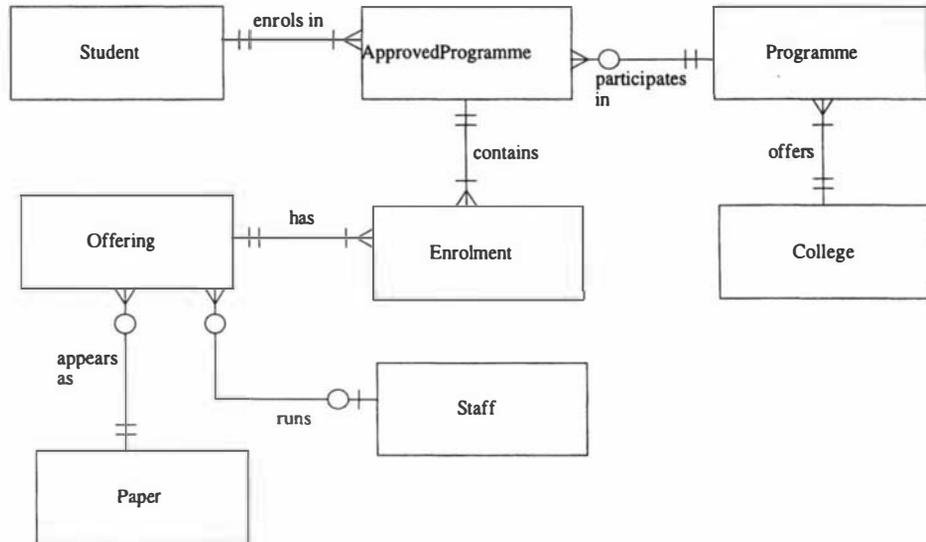
Entity Name	Primary Key	Other Attributes
ApprovedProgramme	<i>Student_id</i> , <i>Programme_code</i>	HoD_approval, Recommended_grade, Start_year
Enrolment	<i>Student_id</i> , <i>Programme_code</i> , <i>Paper_code</i> , <i>Semester</i> , <i>Year</i>	Approved_content, Result, Project_title, <i>Staff_code</i>
Offering	<i>Paper_code</i> , Semester, Year	Points_value, Type, <i>Staff_code</i>
Paper	<i>Paper_code</i>	Title, Points
Programme	<i>Programme_code</i>	Programme_name, Points_value College, PG_admin_name, PG_admin_phone_no
Staff	<i>Staff_code</i>	Staff_name, Research_interest
Student	<i>Student_id</i>	Full_name, Address, Email_address, Phone_no, Enrolled_college, Graduate_status, Position, Previous_qualifications, Work_experience, Research_interest

## 6.2 Final Draft Design Model



Entity Name	Primary Key	Other Attributes
ApprovedProgramme	<i>Student_id,</i> <i>Programme_code</i>	HoD_Approval, Recommended_Grade, Start_year
Enrolment	<i>Student_id, Programme_code,</i> <i>Paper_code, Semester, Year</i>	Approved_Content, Result, Project_title, Supervisor
Offering	<i>Paper_code, Semester,</i> <i>Year</i>	Points_value, Type, <i>Staff_code</i>
Paper	<i>Paper_code</i>	Title, Points
Programme	<i>Programme_code</i>	Programme_name, Points_value College, PG_admin_name, PG_admin_phone_no
Staff	<i>Staff_code</i>	Staff_name, Research_interest
Student	<i>Student_id</i>	Full_name, Address, Phone_no, Email_address Enrolled_college, Graduate_status, Position, Previous_qualifications, Work_experience, Research_interest

### 6.3 Pre-verification Design Model



Entity Name	Primary Key	Other Attributes
ApprovedProgramme	<i>Student_id,</i> <i>Programme_code</i>	HoD_approval, Recommended_grade, Start_year,
Enrolment	<i>Student_id,</i> <i>Programme_code,</i> <i>Paper_code,</i> <i>Semester, Year</i>	Approved_content, Result, Project_title, Supervisor
Offering	<i>Paper_code,</i> <i>Semester, Year</i>	Points_value, <i>Staff_code</i>
Paper	Paper_code	Title, Type
Programme	Programme_code	Programme_name, Required_points, <i>College_name</i>
College	College_name	PG_admin_name, PG_admin_phone_no
Staff	Staff_code	Staff_name, Research_interest
Student	Student_id	Full_name, Address, Phone_no, Email_address, Graduate_status, Position Previous_qualifications, Work_experience, Research_interest



# Appendix 7 - Design Verification

## 7.1 NaLER Sentences

### Primary Key Sentences

- S1. *Each ApprovedProgramme is uniquely identified by Student\_id, Programme\_code.*
- S2. *Each College is uniquely identified by College\_name*
- S1. *Each Enrolment is uniquely identified by Student\_id, Programme\_code, Paper\_code, Semester, Year.*
- S2. *Each Offering is uniquely identified by Paper\_code, Semester, Year.*
- S3. *Each Paper is uniquely identified by Paper\_code.*
- S4. *Each Programme is uniquely identified by Programme\_code.*
- S5. *Each Staff is uniquely identified by Staff\_code.*
- S1. *Each student is uniquely identified by Student\_id*

### Attribute Sentences

- S9. *Each ApprovedProgramme (Student\_id, Programme\_code) must have one HoD\_approval.*
- S10. *Each ApprovedProgramme (Student\_id, Programme\_code) must have one Recommended\_grade.*
- S11. *Each ApprovedProgramme (Student\_id, Programme\_code) must have one Start\_year.*
- S12. *Each College (College\_name) must have one PG\_admin\_name.*
- S13. *Each College (College\_name) must have one PG\_admin\_phone\_no.*
- S14. *Each Enrolment (Student\_id, Programme\_code, Paper\_code, Semester, Year) may have one Approved\_content.*
- S15. *Each Enrolment (Student\_id, Programme\_code, Paper\_code, Semester, Year) must have one Result.*
- S16. *Each Enrolment (Student\_id, Programme\_code, Paper\_code, Semester, Year) may have one Project\_title*
- S17. *Each Enrolment (Student\_id, Programme\_code, Paper\_code, Semester, Year) may have one Supervisor*
- S18. *Each Offering (Paper\_code, Semester, Year) must have one Points\_value.*
- S19. *Each Paper (Paper\_code) must have one Title.*
- S20. *Each Paper (Paper\_code) must have one Type.*
- S21. *Each Programme (Programme\_code) must have one Programme\_name.*
- S22. *Each Staff (Staff\_code) must have one Staff\_name.*
- S23. *Each Student (Student\_id) must have one Full\_name.*
- S24. *Each Student (Student\_id) must have one Address.*
- S25. *Each Student (Student\_id) may have one Phone\_no.*
- S26. *Each Student (Student\_id) may have one Email\_address.*
- S27. *Each Student (Student\_id) must have one Graduate\_status.*
- S28. *Each Student (Student\_id) may have one Position.*
- S29. *Each Student (Student\_id) may have one Previous\_qualifications.*

S30. *Each Student (Student\_id) may have one Work\_experience.*

S31. *Each Student (Student\_id) may have one Research\_interest.*

### Relationship Sentences

S32. *Each College (College\_name) must have one or more Programme (Programme\_code).*

*Each Programme (Programme\_code) must belong to only one College (College\_name).*

S33. *Each Programme (Programme\_code) must participate in one or more ApprovedProgramme (Student\_id, Programme\_code).*

*Each ApprovedProgramme (Student\_id, Programme\_code) must belong to only one Programme (Programme\_code).*

S34. *Each Student (Student\_id) must take one or more ApprovedProgramme (Student\_id, Programme\_code).*

*Each ApprovedProgramme (Student\_id, Programme\_code) must belong to only one Student(Student\_id).*

S35. *Each ApprovedProgramme (Student\_id, Programme\_code) must consist of one or more Enrolment (Student\_id, Programme\_code, Paper\_code, Semester, Year).*

*Each Enrolment (Student\_id, Programme\_code, Paper\_code, Semester, Year) must be part of ApprovedProgramme (Student\_id, Programme\_code).*

S36. *Each Enrolment (Student\_id, Programme\_code, Paper\_code, Semester, Year) must belong to one Offering (Paper\_code, Semester, Year).*

*Each Offering (Paper\_code, Semester, Year) may have one or more Enrolment (Student\_id, Programme\_code, Paper\_code, Semester, Year).*

S37. *Each Offering (Paper\_code, Semester, Year) must consist of only one Paper (Paper\_code).*

*Each Paper (Paper\_code) may participate in one or more Offering (Paper\_code, Semester, Year).*

S38. *Each Offering (Paper\_code, Semester, Year) may be run by only one Staff (Staff\_code).*

*Each Staff(Staff\_code) may run one or more Offering (Paper\_code, Semester, Year).*

### Omitted Sentences

S39. *Each ApprovedProgramme (Student\_id, Programme\_code) must have one Endorsement.*

S40. *Each Programme (Programme\_code) must have one Required\_points.*

S41. *Each Staff (Staff\_code) may have one Research\_interest.*

### Added Sentences

S42. *Each ApprovedProgramme (Student\_id, Programme\_code) may have one Intended\_finish\_year.*

S43. *Each ApprovedProgramme (Student\_id, Programme\_code) may have one College\_approval\_date*

### Replacement Sentence

S44. *Each ApprovedProgramme (Student\_id, Programme\_code) may have one HoD\_approval\_date*

## 7.2 NaLER Example Sentences

**S9. Each ApprovedProgramme (Student\_id, Programme\_code) must have one HoD\_approval.**

ApprovedProgramme (98010101,MSc) must have HoD\_approval 'IS HoD'  
 ApprovedProgramme (98010101,DipSocSc) must have HoD\_approval 'IS HoD'  
 ApprovedProgramme (98020202,MSc) must have HoD\_approval 'IS HoD'  
 ApprovedProgramme (98060606,MSc) must have HoD\_approval 'CS\_HoD'

**View** - College Administrator Fact 11, related to Student Fact 11. (Conflict with HoD Fact 16 and 17)

**Notes** - In the original examples, the full Programme name was used rather than the Programme\_code.

**S10. Each ApprovedProgramme(Student\_id, Programme\_code) must have one Recommended\_grade.**

ApprovedProgramme (98010101,DipSc) has Recommended\_grade 'Distinction'  
 ApprovedProgramme (98010101,MSc) has Recommended\_grade 'First Class Honours'  
 ApprovedProgramme (98020202,MSc) has Recommended\_grade 'Distinction'

**View** - College Administrator Fact 13.

**Notes** - As S9 above.

**S11. Each ApprovedProgramme (Student\_id, Programme\_code) must have one Start\_year.**

ApprovedProgramme (98010101,DipSc) has Start\_year '1997'  
 ApprovedProgramme (98010101,MSc) has Start\_year '1998'  
 ApprovedProgramme (98020202,MSc) has Start\_year '1997'

**View** - None but related to Student Fact 5.

**Notes** - This fact was created during the generation of alternative solutions. It corresponds partly to the Year attribute of the primary key of the ApprovedProgramme entity of the Student view, which was considered unnecessary during the view amalgamation phase. These examples created by the developer.

**S12. Each College (College\_name) must have one PG\_admin\_name.**

College (Business) has PG\_admin\_name 'Alison Gustafson'  
 College (Science) has PG\_admin\_name 'Mike Hardman'.

**View** - None.

**Notes** - This fact was created during the generation of alternative solutions. It provides information on who needs to be notified of Proposed and Completed Programmes. These examples have thus been created by the developer.

**S13. Each College (College\_name) must have one PG\_admin\_phone\_no.**

College (Business) has PG\_admin\_phone\_no '4222'  
 College (Science) has PG\_admin\_phone\_no '4333'.

**View** - None.

**Notes** - This fact was created during the generation of alternative solutions. These examples have thus been created by the developer.

**S14. Each Enrolment (Student\_id, Programme\_code, Paper\_code, Semester, Year) may have one Approved\_content.**

Enrolment (98010101, DipSc, 57.794,1,1997) has Approved-Content '331+ essay'.

Enrolment (98020202, DipSc,57.794,1,1997) has Approved-Content '331+ essay'.

Enrolment (98010101, MSc,57.795,1,1997) has Approved-Content '332+ 341'.

**View** - related to Student Fact 12, HoD Fact 12, Paper Co-ordinator Fact 11

**S15. Each Enrolment (Student\_id, Programme\_code, Paper\_code, Semester, Year) must have one Result.**

Enrolment (98040404, DipInfSc,57.721,1,1998) has Result 'B'.

Enrolment (98040404, DipInfSc,57.794,1,1998) has Result 'A'.

Enrolment (98040404, DipInfSc,57.794,2,1998) has Result 'A'.

**View** - HoD Fact 9, related to College Administrator Fact 1, related to Paper Co-ordinator Fact 10

**S16. Each Enrolment (Student\_id, Programme\_code, Paper\_code, Semester, Year) may have one Project\_title**

Enrolment (98010101, MBS,57.800,12,1997) has Project\_title 'A case for CASE'.

Enrolment (98010101, PhD,57.900,12,1999) has Project\_title 'An investigation into....'.

**View** - HoD Fact 10, related to Paper Co-ordinator Fact 12, related to College Administrator Fact 4

**S17. Each Enrolment (Student\_id, Programme\_code, Paper\_code, Semester, Year) may have one Supervisor**

Enrolment (98010101,DiPc,57.799,2,1996) has Supervisor 'Jon Patrick'.

Enrolment (98010101,MSc,57.800,12,1998) has Supervisor 'Roger Tagg'.

Enrolment (98010101,PhD,57.900,12,1999) has Supervisor 'Jon Patrick'.

Enrolment (98020202, MSc,57.800,12,1998) has Supervisor 'Jon Patrick'.

**View** - related to Student Fact 10, related to College Administrator Fact 5, related to HoD Fact 11

Enrolment (98010101,MSc,57.800,12,1999) has Supervisor 'Richard Whiddett'.

Enrolment (98010101,MSc,57.800,12,1999) has Supervisor 'Chris Freyberg'.

Enrolment (98020202, MSc,57.799,1,1998) has Supervisor 'Richard Whiddett'.

**View** - related to Paper Co-ordinator Fact 13

**Notes** - In this second set of examples the second sentence is now an incorrect example as the decision was taken to hold only the chief supervisor for each project.

**S18. Each Offering (Paper\_code, Semester, Year) must have one Points\_value.**

Offering (57.720,1,1998) has Points\_value '15'.

Offering (57.721,1,1998) has Points\_value '10'.

Offering (57.720,1,1999) has Points\_value '15'.

**View - Paper Co-ordinator Fact 4, related to College Administrator Fact 3, related to Student Fact 5**

**S19. Each Paper (Paper\_code) must have one Title.**

Paper (57.720) has Title 'Information Systems Research Methods'.

Paper (57.722) has Title 'Semantic Modelling'.

Paper (57.794) has Title 'Special Topic in IS'.

Paper (57.795) has Title 'Special Topic in IS'.

**View - Paper Co-ordinator Fact 1**

**S20. Each Paper (Paper\_code) must have one Type.**

Paper (57.794) has Type 'Special topic'.

Paper (57.799) has Type 'Research'.

**View - related to Paper Co-ordinator Fact 3**

**S21. Each Programme (Programme\_code) must have one Programme\_name.**

Programme (MSc) has Programme\_name 'Master of Science'.

Programme (DipInfSci) has Programme\_name 'Diploma of Information Sciences'.

**View - None.**

**Notes -** This sentence was created during the generation of alternatives (Point 1) but is implicit in the use of Programme\_code by three views and Programme\_name by the fourth. These examples were created by the developer.

**S22. Each Staff (Staff\_code) must have one Staff\_name.**

Staff (RT) has the name 'Roger Tagg'.

Staff (JP) has the name 'Jon Patrick'.

**View - None.**

This fact was created during the generation of alternatives (Point 1). These examples here created by the developer.

**S23. Each Student (Student\_id) must have one Full\_name.**

Student (98010101) has the Full\_name 'Michael Robert Ryder'.

Student (98020202) has the Full\_name 'Lisabeth Anne Weston'.

**View - related to Paper Co-ordinator Fact 7, related to Student Fact 8, related to HoD Fact 7, College Administrator Fact 8**

**S24. Each Student (Student\_id) must have one Address.**

Student (98010101) has Address '59 Main Street, Palmerston North'

Student (98020202) has Address '10 Cobden Street, Feilding '

Student (96332211) has Address '59 Main Street, Palmerston North'

**View - Student Fact 6, College Administrator Fact 7, related to Paper Co-ordinator Fact 8**

**S25. Each Student (Student\_id) may have one Phone\_no.**

Student (98010101) has Phone\_no '350-4206'  
 Student (98020202) has Phone\_no '350-5217 '.

**View - Student Fact 9, related to Paper Co-ordinator Fact 8**

**S26. Each Student (Student\_id) may have one Email\_address.**

Student (98010101) has Email\_address 'C.Atkins@massey.ac.nz'  
 Student (98020202) has Email\_address 'L.Weston@massey.ac.nz'

**View - Student Fact 7**

**S27. Each Student (Student\_id) must have one Graduate\_status.**

Student (98040404) has Graduate\_status 'Y'  
 Student (98050505) has Graduate\_status 'Y'

**View - HoD Fact 5**

**S28. Each Student (Student\_id) may have one Position.**

Student (98010101) has Position 'Graduate Assistant - half-time'.  
 Student (98020202) has Position 'Casual Assistant '  
 Student (980030303) has Position 'Graduate Assistant - half-time'.

**View - HoD Fact 3**

**S29. Each Student (Student\_id) may have one Previous\_qualifications.**

Student (98010101) has Previous\_qualifications 'BA(Hons)'.  
 Student (98020202) has Previous\_qualifications 'DipSc '  
 Student (98030303) has Previous\_qualifications 'DipSc , MSc'.

**View - related to HoD Fact 6**

**S30. Each Student (Student\_id) may have one Work\_experience.**

Student (98020202) has Work\_experience '10 years as systems developer '  
 Student (98030303) has Work\_experience '3 years as help desk, 2 years programming'.

**View - none**

**Notes - This fact was created during the generation of alternative solutions (Point 3).**

These examples created by the developer

**S31. Each Student (Student\_id) may have one Research\_interest.**

Student (98040404) has Research\_interest 'conceptual modelling, methodologies '  
 Student (98010101) has Research\_interest 'conceptual modelling '.

**View - related to HoD Fact 8**

**S32. Each College (College\_name) must have one or more Programme (Programme\_code).**

College (Business) has Programme 'BBS (Hons)'.  
 College (Business) has Programme 'PhD'.  
 College (Science) has Programme 'PhD'.

**R. Each Programme (Programme\_code) must belong to only one College (College\_name).**

Programme 'BBS (Hons)' belongs to College (Business)  
 Programme 'PhD' belongs to College (Business)  
 Programme 'PhD' belongs to College (Science)

**View** - related to HoD Fact 15, related to Paper Co-ordinator Fact 9

**Notes** - This fact was created during the normalisation check. The examples were created by the developer. During their creation a potential problem was highlighted in that the PhD program was not unique to a College. In fact an unidentified many to many relationship existed between Programme and College. This needed to be investigated further with the user.

**S33. Each Programme (Programme\_code) must participate in one or more ApprovedProgramme (Student\_id, Programme\_code).**

Programme 'BBS (Hons)' participates in ApprovedProgramme (98010101,BBS(Hons)).

Programme 'PhD' participates in ApprovedProgramme (98010101,PhD)).

Programme 'PhD' participates in ApprovedProgramme (98020202,PhD).

**R. Each ApprovedProgramme (Student\_id, Programme\_code) must belong to only one Programme (Programme\_code).**

Programme 'BBS (Hons)' participates in ApprovedProgramme (98010101,BBS(Hons)).

Programme 'PhD' participates in ApprovedProgramme (98010101,PhD)).

Programme 'PhD' participates in ApprovedProgramme (98020202,PhD).

**View** - none

**Notes** - This fact was created during the generation of alternatives phase (Point 1). These examples were therefore created by the developer.

**S34. Each Student (Student\_id) must take one or more ApprovedProgramme (Student\_id, Programme\_code).**

Student (98010101) takes ApprovedProgramme (98010101,MSc).

Student (98010101) takes ApprovedProgramme (98010101,PhD).

Student (98020202) takes ApprovedProgramme (98010101,MSc).

**Each ApprovedProgramme (Student\_id, Programme\_code) must belong to only one Student (Student\_id).**

ApprovedProgramme (98010101,MSc) belongs to Student (98010101).

ApprovedProgramme (98010101,PhD) belongs to Student (98010101).

ApprovedProgramme (98020202,MSc) belongs to Student (98020202).

**View** - related to Student Fact 5, related to College Administrator Fact 6, related to HoD Fact 4

**S35. Each ApprovedProgramme (Student\_id, Programme\_code) must consist of one or more Enrolment (Student\_id, Programme\_code, Paper\_code, Semester, Year).**

ApprovedProgramme (98010101,MSc) consists of Enrolment (98010101,MSc,  
57.800,12,1998).

ApprovedProgramme (98010101,MSc) consists of Enrolment (98010101,MSc,  
57.720,1,1998).

ApprovedProgramme (98010101,PhD) consists of Enrolment (98010101,PhD,  
57.900,12,1999).

ApprovedProgramme (98020202,MSc) consists of Enrolment (98020202,MSc,  
57.800,12,1998).

**Each Enrolment (Student\_id, Programme\_code, Paper\_code, Semester, Year) must be part of ApprovedProgramme (Student\_id, Programme\_code).**

Enrolment (98010101,MSc, 57.800,12,1998) is part of  
ApprovedProgramme (98010101,MSc).

Enrolment (98010101,MSc, 57.720,1,1998) is part of  
ApprovedProgramme (98010101,MSc).

Enrolment (98010101,PhD, 57.900,12,1999) is part of  
ApprovedProgramme (98010101,PhD).

Enrolment (98020202,MSc,57.800,12,1998) is part of  
ApprovedProgramme (98020202,MSc).

**View** - related to Student Fact 4, related to College Administrator Fact 9, related to HoD Fact 13,

**S36. Each Enrolment (Student\_id, Programme\_code, Paper\_code, Semester, Year) must belong to one Offering (Paper\_code, Semester, Year).**

Enrolment (98040404,DipInfSc, 57.794,1,1998) belong to Offering (57.794,1,1998).

Enrolment (98040404,DipInfSc, 57.794,2,1998) belong to Offering (57.794,2,1998).

Enrolment (98040404,DipInfSc, 57.794,1,1999) belong to Offering (57.794,1,1999).

Enrolment (98050505,DipInfSc, 57.794,1,1998) belong to Offering (57.794,1,1998).

**Each Offering (Paper\_code, Semester, Year) may have one or more Enrolment (Student\_id, Programme\_code, Paper\_code, Semester, Year).**

Offering (57.794,1,1998) has Enrolment (98040404,DipInfSc, 57.794,1,1998).

Offering (57.794,2,1998) has Enrolment (98040404,DipInfSc, 57.794,2,1998).

Offering (57.794,1,1999) has Enrolment (98040404,DipInfSc, 57.794,1,1999).

Offering (57.794,1,1998) has Enrolment (98050505,DipInfSc, 57.794,1,1998).

**View** - related to HoD Fact 18, related to College Administrator Fact 14, related to Paper Co-ordinator Fact 6

**S37. Each Offering (Paper\_code, Semester, Year) must consist of only one Paper (Paper\_code).**

Offering (57.794,1,1998) consists of Paper (57.794).

Offering (57.794,2,1998) consists of Paper (57.794).

Offering (57.794,1,1999) consists of Paper (57.794).

Offering (57.720,1,1998) consists of Paper (57.720).

**Each Paper (Paper\_code) may participate in one or more Offering (Paper\_code, Semester, Year).**

Paper (57.794) participates in Offering (57.794,1,1998).

Paper (57.794) participates in Offering (57.794,2,1998).

Paper (57.794) participates in Offering (57.794,1,1999).

Paper (57.720) participates in Offering (57.720,1,1998).

**View** - related to Paper Co-ordinator Fact 2

**S38. Each Offering (Paper\_code, Semester, Year) may be run by only one Staff (Staff\_code).**

Offering (57.794,1,1998) is run by Staff (CA).

Offering (57.794,2,1998) is run by Staff (JP).

Offering (57.794,1,1999) is run by Staff (CA).

Offering (57.720,1,1998) is run by Staff (LH).

**Each Staff (Staff\_code) may run one or more Offering (Paper\_code, Semester, Year).**

Staff (CA) runs Offering (57.794,1,1998).

Staff (JP) runs Offering (57.794,2,1998).

Staff (CA) runs Offering (57.794,1,1999).

Staff (LH) runs Offering (57.720,1,1998).

**View** - Paper Co-ordinator Fact 5, HoD Fact 1, related to Student Fact 1

### Omitted Sentences

**S39. Each ApprovedProgramme (Student\_id, Programme\_code) must have one Endorsement.**

ApprovedProgramme (98010101,DipSc) has Endorsement 'Computing'  
 ApprovedProgramme (98010101,MSc) has Endorsement 'Information Systems'  
 ApprovedProgramme (98020202,MSc) has Endorsement 'Information Systems'

**View** - None but related to Student Fact 5.

**S40. Each Programme (Programme\_code) must have one Required\_points.**

Programme (MBS) has Required\_points '200'  
 Programme (DipInfSci) has Required\_points '90'  
 Programme (MSc) has Required\_points '200'

**View** - Student Fact 3.

**S41. Each Staff (Staff\_code) may have one Research\_interest.**

Staff (CA) has Research\_interest 'Conceptual Modelling, CASE tools'  
 Staff (CF) has Research\_interest 'Conceptual Modelling'  
 Staff (MR) has Research\_interest 'Conceptual Modelling, CASE tools'

**View** - HoD Fact 2, related to Paper Coord Fact 14.

### New sentences

**S42. Each ApprovedProgramme (Student\_id, Programme\_code) may have one Intended\_finish\_year.**

ApprovedProgramme (98010101,DipSc) has Intended\_finish\_year '1998'  
 ApprovedProgramme (98010101,MSc) has Intended\_finish\_year '2000'  
 ApprovedProgramme (98020202,MSc) has Intended\_finish\_year '1998'

**View** - None but related to HoD Fact 14

**S43. Each ApprovedProgramme (Student\_id, Programme\_code) may have one College\_approval**

ApprovedProgramme (98010101,DipSc) has College\_approval\_date '21-5-1997'  
 ApprovedProgramme (98010101,MSc) has College\_approval\_date '21-5-1998'  
 ApprovedProgramme (98020202,MSc) has College\_approval\_date '21-5-1998'

**View** - None but related to HoD Fact 16

### Replacement Sentences

**S44. Each ApprovedProgramme (Student\_id, Programme\_code) must have one HoD\_approval\_date.**

ApprovedProgramme (98010101,MSc) has HoD\_approval\_date '1-3-1998'  
 ApprovedProgramme (98010101,DipSocSc) has HoD\_approval\_date '1-3-1998'  
 ApprovedProgramme (98020202,MSc) has HoD\_approval\_date '28-2-1998'

**View** - related to College Administrator Fact 11, related to Student Fact 11, related to HoD Fact 17

**Notes** - Replaces S9

### Late Addition

**S45. Each Offering (Paper\_code, Semester, Year) must be offered in one Mode.**

Offering (57.720,1,1999) is offered 'Internal'  
Offering (57.720,1,2000) is offered 'Internal'  
Offering (57.723,2,1999) is offered 'Block'

# Appendix 8 - Equivalence Tables

## 8.1 NaLER Sentences v Infological Analysis Model

View	Fact	Sentence	Notes	Action
<b>Student</b>	1	38		
	2	18		
	3		missing	added, S 40
	4	35		
	5	11/34	but endorsement missing	added, S 39
	6	24		
	7	26		
	8	23		
	9	25		
	10	17		
	11	9		
	12	14		

View	Fact	Sentence	Notes	Action
<b>Paper Coord</b>	1	19		
	2	37		
	3	20		
	4	18		
	5	38		
	6	36		
	7	23		
	8	24/25		
	9	32		
	10	15		
	11	14		
	12	16		
	13	17		
	14		missing	now related to S 41

View	Fact	Sentence	Notes	Action
<b>HoD</b>	1	38		
	2		missing	added, S41
	3	28		
	4	34		
	5	27		
	6	29		
	7	23		
	8	31		
	9	15		
	10	16		
	11	17		
	12	14		
	13	35		
	14		missing	added Intended_finish_year, S42
	15	32		
	16		conflict with S9	added College_approval_date, S43
	17		conflict with S9	renamed HoD_approval_date, S44
	18	36		

View	Fact	Sentence	Notes	Action
<b>College Administrator</b>	1	15		
	2		derived fact	
	3	18		
	4	16		
	5	17		
	6	34		
	7	24		
	8	23		
	9	35		
	10		derived fact	
	11	9		
	12		derived fact	
	13	10		
	14	36		

## 8.2 Verification Views

### 8.2.1 Student

F#	S#	Sentences
1	38	<p><i>Each Offering(Paper_code,Semester, Year) may be run by only one Staff(Staff_code)</i></p> <p><i>Each Staff(Staff_code)may run one or more Offering(Paper_code,Semester, Year).</i></p>
2	18	<i>Each Offering(Paper_code,Semester,Year) must have one Points_value.</i>
3	40	<i>Each Programme(Programme_code) must have one Required_points.</i>
4	35	<p><i>Each ApprovedProgramme (Student_id,Programme_code) must consist of one or more Enrolment(Student_id,Programme_code, Paper_code, Semester, Year).</i></p> <p><i>Each Enrolment(Student_id,Programme_code, Paper_code, Semester, Year) must be part of ApprovedProgramme (Student_id,Programme_code).</i></p>
5	11 34	<p><i>Each ApprovedProgramme(Student_id,Programme_code) must have one Start_year.</i></p> <p><i>Each Student(Student_id) must take one or more ApprovedProgramme (Student_id, Programme_code).</i></p> <p><i>Each ApprovedProgramme (Student_id,Programme_code) must belong to only one Student(Student_id).</i></p>
6	24	<i>Each Student(Student_id) must have one Address.</i>
7	26	<i>Each Student(Student_id) may have one Email_address.</i>
8	23	<i>Each Student(Student_id) must have one Full_name.</i>
9	25	<i>Each Student(Student_id) may have one Phone_no.</i>
10	17	<i>Each Enrolment(Student_id,Programme_code.Paper_code,Semester,Year) may have one Supervisor</i>
11	44	<i>Each ApprovedProgramme(Student_id,Programme_code) may have one HoD_approval_date.</i>
12	14	<i>Each Enrolment(Student_id,Programme_code.Paper_code,Semester,Year) may have one Approved_content.</i>

### 8.2.2 Paper Co-ordinator View

F#	S#	Examples
1	19	<i>Each Paper(Paper_code) must have one Title.</i>
2	37	<i>Each Offering(Paper_code,Semester, Year) must consist of only one Paper(Paper_code). Each Paper(Paper_code) may participate in one or more Offering(Paper_code,Semester, Year).</i>
3	20	<i>Each Paper(Paper_code) must have one Type.</i>
4	18	<i>Each Offering(Paper_code,Semester,Year) must have one Points_value.</i>
5	38	<i>Each Offering(Paper_code,Semester, Year) may be run by only one Staff(Staff_code).  Each Staff(Staff_code)may run one or more Offering(Paper_code,Semester, Year).</i>
6	36	<i>Each Enrolment(Student_id,Programme_code, Paper_code, Semester, Year) must belong to one Offering(Paper_code,Semester, Year).  Each Offering(Paper_code,Semester, Year) may have one or more Enrolment (Student_id,Programme_code, Paper_code, Semester, Year).</i>
7	23	<i>Each Student(Student_id) must have one Full_name.</i>
8	24/ 25	<i>Each Student(Student_id) must have one Address. Each Student(Student_id) may have one Phone_no.</i>
9	32	<i>Each College(College_name) must have one or more Programme(Programme_code). Each Programme(Programme_code) must belong to only one College(College_name).</i>
10	15	<i>Each Enrolment(Student_id,Programme_code.Paper_code,Semester,Year) must have one Result.</i>
11	14	<i>Each Enrolment(Student_id,Programme_code.Paper_code,Semester,Year) may have one Approved_content.</i>
12	16	<i>Each Enrolment(Student_id,Programme_code.Paper_code,Semester,Year) may have one Project_title</i>
13	17	<i>Each Enrolment(Student_id,Programme_code.Paper_code,Semester,Year) may have one Supervisor</i>
14	41	<i>Each Staff(Staff_code) may have one Research_interest.</i>

### 8.2.3 Head of Department View

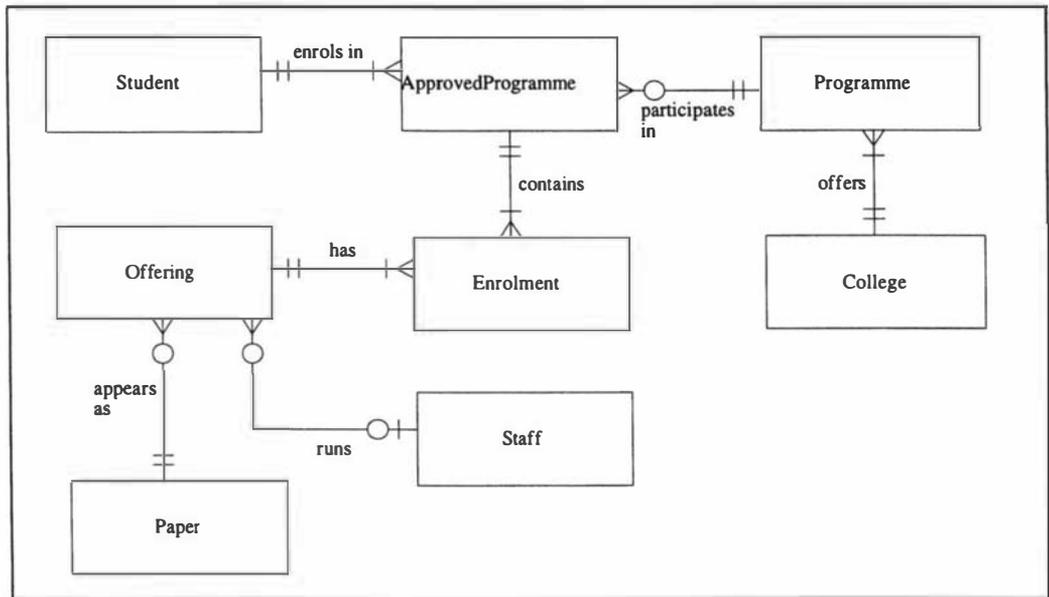
F#	S#	Examples
1	38	<p><i>Each Offering(Paper_code,Semester, Year) may be run by only one Staff(Staff_code).</i></p> <p><i>Each Staff(Staff_code)may run one or more Offering (Paper_code,Semester, Year).</i></p>
2	41	<i>Each Staff(Staff_code) may have one Research_interest.</i>
3	28	<i>Each Student(Student_id) may have one Position.</i>
4	34	<p><i>Each Student(Student_id) must take one or more ApprovedProgramme (Student_id,Programme_code).</i></p> <p><i>Each ApprovedProgramme (Student_id,Programme_code) must belong to only one Student(Student_id).</i></p>
5	27	<i>Each Student(Student_id) must have one Graduate_status.</i>
6	29	<i>Each Student(Student_id) may have one Previous_qualifications.</i>
7	23	<i>Each Student(Student_id) must have one Full_name.</i>
8	31	<i>Each Student(Student_id) may have one Research_interest.</i>
9	15	<i>Each Enrolment(Student_id,Programme_code.Paper_code,Semester,Year) must have one Result.</i>
10	16	<i>Each Enrolment(Student_id,Programme_code.Paper_code,Semester,Year) may have one Project_title</i>
11	17	<i>Each Enrolment(Student_id,Programme_code.Paper_code,Semester,Year) may have one Supervisor</i>
12	14	<i>Each Enrolment(Student_id,Programme_code.Paper_code,Semester,Year) may have one Approved_content.</i>
13	35	<p><i>Each ApprovedProgramme (Student_id,Programme_code) must consist of one or more Enrolment(Student_id,Programme_code, Paper_code, Semester, Year).</i></p> <p><i>Each Enrolment(Student_id,Programme_code, Paper_code, Semester, Year) must be part of ApprovedProgramme (Student_id,Programme_code).</i></p>
14	42	<i>Each ApprovedProgramme(Student_id,Programme_code) may have one Intended_finish_year.</i>
15	32	<p><i>Each College(College_name) must have one or more Programme(Programme_code).</i></p> <p><i>Each Programme(Programme_code) must belong to only one College(College_name).</i></p>
16	43	<i>Each ApprovedProgramme(Student_id,Programme_code) may have one College_approval_date</i>
17	44	<i>Each ApprovedProgramme(Student_id,Programme_code) may have one HoD_approval_date</i>
18	36	<p><i>Each Enrolment(Student_id,Programme_code, Paper_code, Semester, Year) must belong to one Offering(Paper_code,Semester, Year).</i></p> <p><i>Each Offering(Paper_code,Semester, Year) may have one or more Enrolment (Student_id,Programme_code, Paper_code, Semester, Year).</i></p>

### 8.2.4 College Administrator View

F#	S#	Examples
1	15	<i>Each</i> Enrolment(Student_id,Programme_code,Paper_code,Semester,Year) <i>must have one</i> Result.
2		derived - Enrolment.result(value) *Offering.Points = grade point score
3	18	<i>Each</i> Offering(Paper_code,Semester,Year) <i>must have one</i> Points_value.
4	16	<i>Each</i> Enrolment(Student_id,Programme_code,Paper_code,Semester,Year) <i>may have one</i> Project_title
5	17	<i>Each</i> Enrolment(Student_id,Programme_code,Paper_code,Semester,Year) <i>may have one</i> Supervisor
6	34	<i>Each</i> Student(Student_id) <i>must take one or more</i> ApprovedProgramme (Student_id,Programme_code).  <i>Each</i> ApprovedProgramme (Student_id,Programme_code) <i>must belong to only one</i> Student(Student_id).
7	24	<i>Each</i> Student(Student_id) <i>must have one</i> Address.
8	23	<i>Each</i> Student(Student_id) <i>must have one</i> Full_name.
9	35	<i>Each</i> ApprovedProgramme (Student_id,Programme_code) <i>must consist of one or more</i> Enrolment(Student_id,Programme_code, Paper_code, Semester, Year).  <i>Each</i> Enrolment(Student_id,Programme_code, Paper_code, Semester, Year) <i>must be part of</i> ApprovedProgramme (Student_id,Programme_code).
10		derived fact - grade point score/Enrolment.Offering.Points(sum)
11	44	<i>Each</i> ApprovedProgramme(Student_id,Programme_code) <i>may have one</i> HoD_approval_date
12		derived fact
13	10	<i>Each</i> ApprovedProgramme(Student_id,Programme_code) <i>must have one</i> Recommended_Grade.
14	36	<i>Each</i> Enrolment(Student_id,Programme_code, Paper_code, Semester, Year) <i>must belong to one</i> Offering(Paper_code,Semester, Year).  <i>Each</i> Offering(Paper_code,Semester, Year) <i>may have one or more</i> Enrolment (Student_id,Programme_code, Paper_code, Semester, Year).

# Appendix 9 - Design Documentation

## 9.1 Verified Design Model - Datalogical



Entity Name	Primary Key	Other Attributes
ApprovedProgramme	<i>Student_id,</i> <i>Programme_code</i>	HoD_approval_date, College_approval_date, Recommended_grade, Start_year, Intended_finish_year Endorsement
Enrolment	<i>Student_id,</i> <i>Programme_code,</i> <i>Paper_code,</i> <i>Semester, Year</i>	Approved_content, Result, Project_title, Supervisor
Offering	<i>Paper_code,</i> <i>Semester, Year</i>	Points_value, <i>Staff_code</i>
Paper	<i>Paper_code</i>	Title, Type
Programme	<i>Programme_code</i>	Programme_name, Required_points, <i>College_name</i>
College	<i>College_name</i>	PG_admin_name, PG_admin_phone_no
Staff	<i>Staff_code</i>	Staff_name, Research_interest
Student	<i>Student_id</i>	Full_name, Address, Phone_no, Email_address, Graduate_status, Position Previous_qualifications, Work_experience,

		Research_interest
--	--	-------------------

## 9.2 Relational Schema

```

CREATE TABLE ApprovedProgramme (
    Student_id          smallint          NOT NULL,
    Programme_code     char varying(20) NOT NULL,
    HoD_approval_date  date              NULL,
    College_approval_date date          NULL,
    Start_year         smallint          NOT NULL,
    Recommended_grade  char (3) NULL,
    Endorsement        char varying(30) NULL,
    Intended_finish_year smallint        NULL,

    PRIMARY KEY (Student_id, Programme_code),
    FOREIGN KEY (Student_id)      REFERENCES Student,
    FOREIGN KEY (Programme_code)  REFERENCES Programme
)

```

```

CREATE TABLE College (
    College_name      char (20)          NOT NULL,
    PG_admin_name    char (30)          NULL,
    PG_admin_phone_no char (4)          NULL,

    PRIMARY KEY (College_name)
)

```

```

CREATE TABLE Enrolment (
    Student_id      smallint          NOT NULL,
    Programme_code  char (20)         NOT NULL,
    Paper_code      char (6)          NOT NULL,
    Semester        smallint          NOT NULL,
    Year            smallint          NOT NULL,
    Approved_content varchar (60)     NULL,
    Result          char(6)           NULL,
    Project_title   varchar(80) NULL,
    Supervisor      varchar(30) NULL,

    PRIMARY KEY (Student_id, Programme_code, Paper_code, Semester, Year),
    FOREIGN KEY (Student_id, Programme_code) REFERENCES ApprovedProgramme,
    FOREIGN KEY (Paper_code, Semester, Year)  REFERENCES Offering
)

```

```

CREATE TABLE Offering (
    Paper_code      char (6)          NOT NULL,
    Semester        smallint          NOT NULL,
    Year            smallint          NOT NULL,
    Staff_code      char (3)          NULL,
    Points_value    smallint          NULL,

    PRIMARY KEY (Paper_code, Semester, Year),
    FOREIGN KEY (Paper_code)      REFERENCES Paper,
    FOREIGN KEY (Staff_code)      REFERENCES Staff )

```

```

CREATE TABLE Paper (
    Paper_code      char(6)          NOT NULL,
    Title           varchar(20)      NOT NULL,
)

```

Type char(2) NOT NULL,

PRIMARY KEY (Paper\_code)

)

CREATE TABLE Programme (

Programme\_code char(20) NOT NULL,  
 College\_name char(20) NOT NULL,  
 Programme\_name varchar(40) NOT NULL,  
 Required\_points smallint NOT NULL,

PRIMARY KEY (Programme\_code),

FOREIGN KEY (College\_name) REFERENCES College

)

CREATE TABLE Staff (

Staff\_code char(3) NOT NULL,  
 Staff\_name char(30) NOT NULL,  
 Staff\_research\_interests varchar(80) NULL,

PRIMARY KEY (Staff\_code)

)

CREATE TABLE Student (

Student\_id smallint NOT NULL,  
 Full\_name char(30) NULL,  
 Address varchar(50) NULL,  
 Phone\_no char(15) NULL,  
 Email varchar(30) NULL,  
 Graduate\_status logical NULL,  
 Position varchar(20) NULL,  
 Previous\_qualifications varchar(40) NULL,  
 Work\_experience varchar(80) NULL,  
 Research\_interest varchar(80) NULL,

PRIMARY KEY (Student\_id)

)

# Appendix 10 - Verified Design Model

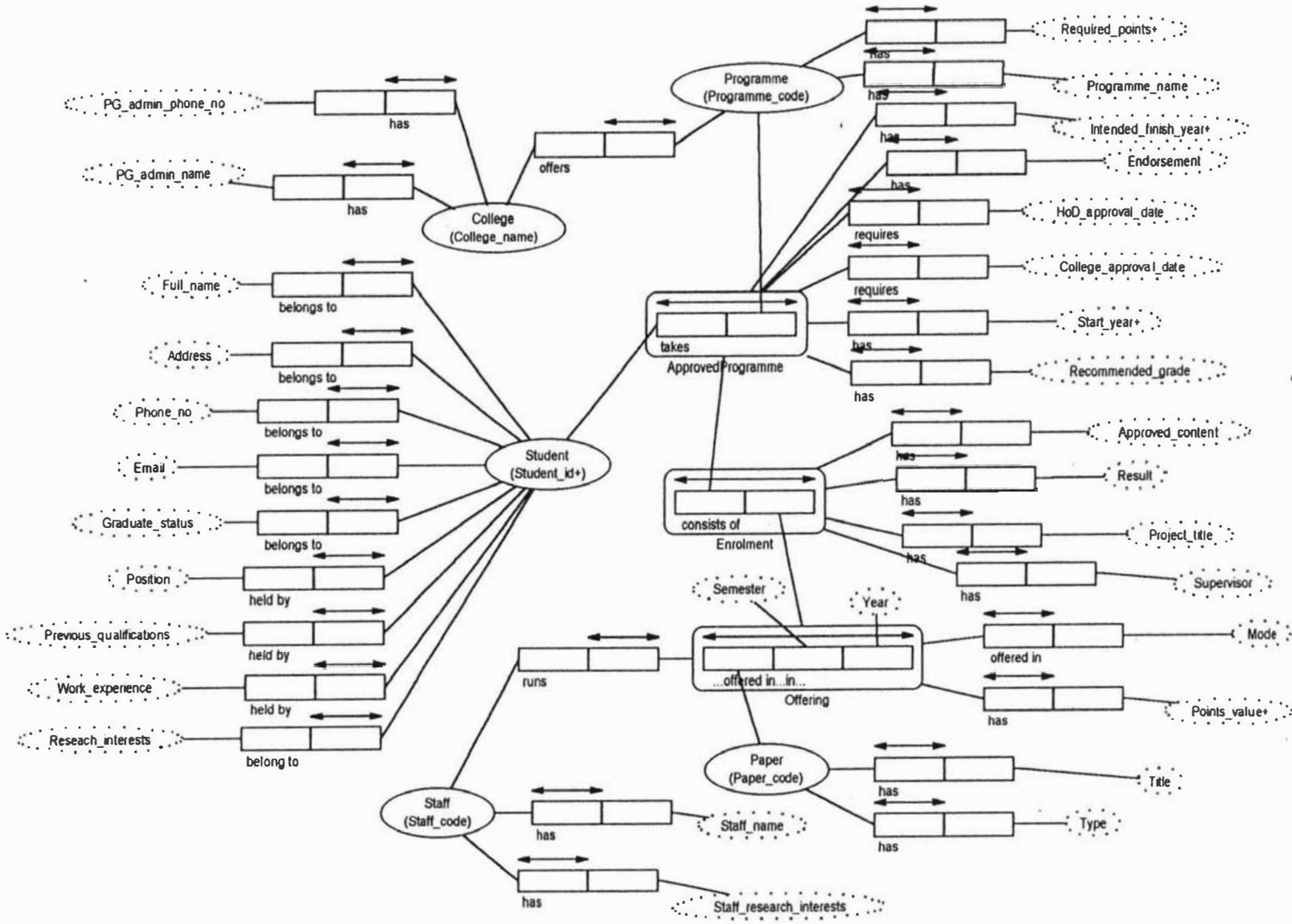
## Infological

### 10.1 Fact Sentences

1. Address belongs to Student  
Each Student has at most one Address that belongs to it
2. ApprovedProgramme consists of Offering  
Each ApprovedProgramme consists of zero or more Offering and  
Each Offering has zero or more ApprovedProgramme that consists of it
3. ApprovedProgramme has Endorsement  
Each ApprovedProgramme has at most one Endorsement
4. ApprovedProgramme has Intended\_finish\_year  
Each ApprovedProgramme has at most one Intended\_finish\_year
5. ApprovedProgramme has Recommended\_grade  
Each ApprovedProgramme has at most one Recommended\_grade
6. ApprovedProgramme has Start\_year  
Each ApprovedProgramme has at most one Start\_year
7. ApprovedProgramme requires College\_approval\_date  
Each ApprovedProgramme requires at most one College\_approval\_date
8. ApprovedProgramme requires HoD\_approval\_date  
Each ApprovedProgramme requires at most one HoD\_approval\_date
9. College has PG\_admin\_name  
Each College has at most one PG\_admin\_name
10. College has PG\_admin\_phone\_no  
Each College has at most one PG\_admin\_phone\_no
11. College offers Programme  
Each Programme has at most one College that offers it
12. Email belongs to Student  
Each Student has at most one Email that belongs to it
13. Enrolment has Approved\_content  
Each Enrolment has at most one Approved\_content
14. Enrolment has Project\_title  
Each Enrolment has at most one Project\_title
15. Enrolment has Result  
Each Enrolment has at most one Result

16. Enrolment has Supervisor  
Each Enrolment has at most one Supervisor
17. Full\_name belongs to Student  
Each Student has at most one Full\_name that belongs to it
18. Graduate\_status belongs to Student  
Each Student has at most one Graduate\_status that belongs to it
19. Offering has Points\_value  
Each Offering has at most one Points\_value
20. Offering offered in Mode  
Each Offering offered in at most one Mode
21. Paper has Title  
Each Paper has at most one Title
22. Paper has Type  
Each Paper has at most one Type
23. Paper offered in Semester in Year  
Each Paper, Semester, Year combination is unique
24. Phone\_no belongs to Student  
Each Student has at most one Phone\_no that belongs to it
25. Position held by Student  
Each Student has at most one Position that held by it
26. Previous\_qualifications held by Student  
Each Student has at most one Previous\_qualifications that held by it
27. Programme has Programme\_name  
Each Programme has at most one Programme\_name
28. Programme has Required\_points  
Each Programme has at most one Required\_points
29. Reseach\_interests belong to Student  
Each Student has at most one Reseach\_interests that belong to it
30. Staff has Staff\_name  
Each Staff has at most one Staff\_name
31. Staff has Staff\_research\_interests  
Each Staff has at most one Staff\_research\_interests
32. Staff runs Offering  
Each Offering has at most one Staff that runs it
33. Student takes Programme  
Each Student takes zero or more Programme and  
Each Programme has zero or more Student that takes it
34. Work\_experience held by Student  
Each Student has at most one Work\_experience that held by it

10.2 ORM Diagram





# Appendix 11 - Design Innovation

## Comparison of NaLER sentences to Fact Types

Sentence	Final Design	Student	Paper Coord	HoD	College Admin
9	replaced by S44	S11			C11
10	5				C13
11	6	S5			
12 new	9				
13 new	10				
14	13	S12	P11	H12	
15	15		P10	H9	C1
16	14		P12	H10	C4
17	16	S10	P13	H11	C5
18	19	S2	P4		C3
19	21		P1		
20	22		P3		
21 new	27				
22 new	30				
23	17	S8	P7	H7	C8
24	9	S6	P8		C7
25	24	S9	P8		
26	12	S7			
27	18			H5	
28	25			H3	
29	26			H6	
30 new	34				
31	29			H8	
32	11		P9	H15	
33 new	33				
34	33	S5		H4	C6
35	2	S4		H13	C9
36	2		P6	H18	C14
37	23		P2		
38	32	S1	P5	H1	
39	3	S5			
40	28	S3			
41	31		P14	H2	
42	4			H14	
43	7			H16	
44	8			H17	
45 new	20				

The NaLER sentences recorded in the highlighted rows cannot be traced back to any analysis Fact Types. They are thus termed 'innovative'.



## Appendix 12 - Task Checklists

### Analysis Stage

No	Task	Comments	Date	Quality goal
<b>1</b>	<b>Record system expectation</b>			<b>A6</b>
1.1	Context Diagram			
1.2	Problem Definition			
1.3	User Requests			
1.4	Other			
<b>2</b>	<b>Identify appropriate users</b>			
<b>3</b>	<b>Initial requirement collection</b>	for each user		
3.1	Interview			
3.2	Documents			
3.3	Other			
3.4	Initial sentences confirmed	each user		<b>A1, A2</b>
<b>4</b>	<b>Construction of qualified fact types</b>			
<b>5</b>	<b>Confirmation of qualified fact types</b>	each user		<b>A1, A2</b>
<b>6</b>	<b>Collection of examples</b>			
<b>7</b>	<b>Confirmation of example sentences</b>	each user		<b>A1, A2</b>
<b>8</b>	<b>Syntactic verification</b>			<b>A5</b>
<b>9</b>	<b>Expectation matching</b>	each user		<b>A6</b>
<b>10</b>	<b>Confirmation that sentences are understandable and understood</b>	each user		<b>A3</b>
<b>11</b>	<b>Task checklist complete</b>			<b>A4</b>

### Design Stage

No	Task	Comments	Date	Quality Goal
<b>1</b>	<b>Prepare first draft design model</b>			
1.1	Transform analysis models to relational representations			
1.2	Amalgamate logical views			
1.2.1	List all entities and PKs			
1.2.2	Merge entities with same PK			
1.2.3	Check for synonyms			
1.2.4	Check for similar PKs			
<b>2</b>	<b>Generate/evaluate alternatives</b>			
<b>3</b>	<b>Incorporate future requirements</b>			
<b>4</b>	<b>Create final draft design</b>			
<b>5</b>	<b>Verify final design</b>			
5.1	Syntax Check			<b>D10</b>
5.1.1	Create 2-way sentences			
5.1.2	Check participation constraints			
5.1.3	Check PK-FK links			
5.1.4	Check normalisation			<b>D8</b>
5.2	Simplicity check			<b>D9</b>
5.2.1	Check for minimal primary keys			
5.2.2	Check for redundant relationships			
5.2.3	Check for trivial relations			
5.2.4	Check all relations are required			
5.3	Semantic check			<b>D2, D3, D7</b>
5.3.1	Create NaLER sentences			
5.3.2	Populate NaLER with examples taken from analysis model			
5.3.3	Create cross reference table – analysis facts to NaLER sentences			
5.3.4	Check for completeness			
5.3.5	Check for consistency			
<b>6</b>	<b>Audit</b>			<b>D6</b>
6.1	Check source of all NaLER sentences			
<b>7</b>	<b>Peer Review</b>			
<b>8.</b>	<b>User Verification</b>			<b>D1, D4</b>
7.1	Create NaLER user views			
7.2	Correlate NaLER and analysis views			
7.3	Gain user verification			
<b>9</b>	<b>Task Checklist complete</b>			<b>D5</b>